

Forward

June 22, 1990

Welcome to the revised version of the U.C. Berkeley Computer Science *Preliminary Exams Study Guide*. This guide is organized into four sections, each sold separately to make it easier for people to obtain only the sections they need. The first three sections are for the three core exams (hardware, software, and theory). Each one contains the ten most recent exams, arranged in ascending chronological order, along with solutions for seven of them. Also included is the most recent version of the exam syllabus, supplied by the faculty. The fourth section contains the syllabi for the oral research area exams along with sample questions reported by students.

The solutions provided are not official (although in a few cases they are the key provided by the faculty members who wrote the exams). Most of the solutions are photocopies of high-scoring student answers. Where possible, the answers photocopied are ones that received full credit. Keep in mind that these answers are taken from unusually high scoring exams, and passing answers need not always be as thorough as those provided here.

If you have questions about one of the exams or general questions about prelims, you can consult the *EECS Graduate Information* booklet, Kathryn Crabtree, or the faculty member in charge of prelims (currently Brian Barsky). Specific questions about the exams can be answered by your fellow students in the Prelim Review sessions which the CSGSA organizes at the beginning of each semester.

Our thanks go to the anonymous students who agreed to contribute their solutions, to David Gedye, who put together the first modern version of this guide, and to Joe Konstan and Steve Lucco, who kept the guide up-to-date during their tenures as CSGSA Prelim Liaison Officers.

Marti Hearst (CSGSA Prelim Liaison Officer)

Kathryn Crabtree (CS Graduate Assistant and Prelims Coordinator)

...

- 3.2.3 Hierarchies
Mano: 12-3; Hamacher: 8.6, 8.7
 - 3.2.3.1 Caches (direct mapping, set associative)
Mano: 12-6; Hamacher: 8.6
 - 3.2.3.2 Virtual memory (how? why?)
Mano: 12-5, 12-7; Hamacher: 8.7
- 3.3 Input/Output
Mano: 8, 11; Hamacher: 8, 9.1, 9.2
 - 3.3.1 Devices (disks, terminals, etc.)
Mano: 11-1; Hamacher: 9.1, 9.2
 - 3.3.2 I/O techniques (e.g., advantages/disadvantages)
Mano: 11-2 to 11-8; Hamacher: 8.2 to 8.5
 - 3.3.2.1 Polling/interrupts
Mano: 11-5; Hamacher: 8.2 to 8.4
 - 3.3.2.2 DMA/data channels
Mano: 11-4, 11-6; Hamacher: 8.2.1, 8.5
 - 3.3.2.3 Priorities
Mano: 11-5; Hamacher: 8.4.4 to 8.4.6
 - 3.3.2.4 Program controlled transfer
Mano: 8-8, 11-2, 11-3; Hamacher: 8.2
- 4. Machine Organization
Mano: 4, 5, 6, 7, 8, 11; Hamacher: 1, 2, 3, 4, 5
 - 4.1 Block diagrams, interactions (busses, I/O organization)
Mano: 5, 7, 11
 - 4.2 Register transfer languages, ISP
Mano: 4
 - 4.2.1 Differences from programming languages
Mano: 4-1
 - 4.2.2 Examples (ISP)
Mano: 4; Siewirook: 4.2.3
 - 4.3 Instruction set design
Mano: 5, 6, 7; Hamacher: 1.3, 2.1 to 2.8, 3.1 to 3.5, 4.1.6
 - 4.3.1 0/1/2/3 address designs
Mano: 7-4; Hamacher: 1.3, 2.3, 2.7, 4.1.6
 - 4.3.2 Addressing modes (immediate, indirect, absolute, etc.)
Mano: 5-3, 5-4, 7-4, 7-5; Hamacher: 2.4, 2.5
 - 4.3.3 Basic operations (e.g., ADD, JUMP)
Mano: 4-2 to 4-7, 5-2, 6-1, 7-6, 7-7; Hamacher: 2.8
 - 4.3.4 Number of registers (≤ 0)
Mano: 4-2, 7-1, 7-4; Hamacher: 2.1
 - 4.3.5 Word size
Mano: 2-6; Hamacher: 2.1
 - 4.3.6 Data types
Mano: 3; Hamacher: 2.1
 - 4.4 Control design (microcode, hardwire)
Mano: 4, 5, 8; Hamacher: 4.3 to 4.5, 5
 - 4.4.1 Hardwired (advantages/disadvantages)
Mano: 4-6, 5-3, 8-7; Hamacher: 4.3
 - 4.4.2 Microcode
Mano: 8; Hamacher: 5
 - 4.4.2.1 Advantages/disadvantages
Mano: 8-7
 - 4.4.2.2 Minimal encoding/maximal encoding (horizontal vs. vertical)
Mano: 8-5
 - 4.4.2.3 Two-level control store
Mano: 8-5
 - 4.4.2.4 Instruction decoding/micro-sequencing
Mano: 8-1 to 8-4
 - 4.4.2.5 Emulation
Mano: 8-7
 - 4.4.2.6 Writable control store
Mano: 8-7

**Computer Science Preliminary Exam
Hardware Syllabus**

- M. Mano, Computer System Architecture, Prentice-Hall, 1982.
- V. Hamacher, Z. Vranesic, S. Zaky, Computer Organization, McGraw-Hill, 1978.
- D. Siewiorek, C. Bell, A. Newell, Computer Structures: Principles and Examples, McGraw-Hill, 1982.
- H. Taub, Digital Circuits and Microprocessors, McGraw-Hill, 2nd Edition, 1982.

1. Machine Arithmetic and Data Representation

Mano: 1, 3, 9, 10; Hamacher: 7, A

1.1 Boolean algebra (e.g., deMorgan's law)

Mano: 1-1 to 1-3; Hamacher: A.3, A.4

1.2 Integer representation (signed magnitude, 1's complement, 2's complement)

Mano: 3-1, 3-2; Hamacher: 7.1

1.3 Floating point representation (e.g., excess encoding, base)

Mano: 3-3, 3-4; Hamacher: 7.10

1.4 Machine arithmetic

Mano: 9, 10; Hamacher: 7.2 to 7.10

1.4.1 Integer arithmetic

Mano: 9-1 to 9-5, 10-1 to 10-3; Hamacher: 7.2 to 7.9

1.4.1.1 Simple operations (e.g., add, subtract, complement)

Mano: 9-1 to 9-3, 10-1, 10-2

1.4.1.2 Multiply/Divide (e.g., Booth's, restoring/non-restoring)

Mano: 9-4, 9-5, 10-3

1.4.2 Floating point

Mano: 10-4; Hamacher: 7.10

1.4.2.1 Add, subtract, multiply, divide

Mano: 10-4

1.4.2.2 Normalization

Mano: 10-4

2. Logic Design

Mano: 1, 2; Hamacher: A

2.1 Logic gates and technologies (why NAND)

Mano: 1-1, 2-1; Hamacher: A.1, A.2, A.4

2.2 Flipflops/registers (master/slave, edge trigger)

Mano: 1-5, 2-2; Hamacher: A.6 to A.10

2.3 Finite state machines

Mano: 1-6; Taub: 7.5

2.4 Programmed logic arrays (PLAs vs. ROMs. Why each)

Mano: 2-7; Hamacher: A.12

2.5 Design techniques

Mano: 1, 2-2 to 2-5; Hamacher: A.3

2.5.1 Combinatorial (e.g., Karnaugh maps)

Mano: 1-2 to 1-4, 2-3; Hamacher: A.3

2.5.2 Sequential (e.g., state minimization)

Mano: 1-5 to 1-7, 2-2, 2-4, 2-5; Hamacher: A.3

3. Machine Components

Mano: 2, 4, 5, 7, 11, 12; Hamacher: 1.3, 4.2, 6, 8, 9

3.1 Processor

Mano: 4, 5, 7; Hamacher: 1.3, 4.2, 8

3.1.1 Microarchitecture (1/2/3 bus architectures)

Mano: 4-1, 4-2, 7-1, 7-8; Hamacher: 1.3

3.1.2 Arithmetic units (carry look-ahead, added vs. ALU)

Mano: 7-2; Hamacher: 4.2, 7.3

3.1.3 Shifters (1 bit, variable bit)

Mano: 7-2

3.2 Memory

Mano: 2, 7, 12; Hamacher: 8.2 to 8.7

3.2.1 RAM/ROM (when used)

Mano: 2-6, 2-7, 12-2; Hamacher: 8.2, 8.3

3.2.2 Organizations (physical organizations, interleaving)

Mano: 7-9, 12; Hamacher: 8.5

University of California
College of Engineering
Department of Electrical Engineering
and Computer Sciences
Computer Science Division
HARDWARE PRELIM EXAM
Spring 1990

General Instructions:

- The exam lasts 3 hours (180 minutes).
- The exam has 17 pages.
- Do all your work on these papers; use backsides if necessary.
- Calculators are not allowed.
- Read the problem statements carefully, at least twice.
- You should not need to ask questions.
- If something seems unclear, state your assumptions.
- The indicated points give you an idea how long a problem should take (minutes).
- There are 10 problems adding up to 180 points. Completing 160 points will be regarded as a perfect score, though you can attempt to score higher.

Please write your id. no. on
every page of the exam.

1. (10 points)

a.(4 points) Give the truth table for a single stage of a Full Binary Adder.

b.(3 points) Express the Sum function and Carry Out function into standard sum of products form, and product of sums form.

$$S = x \oplus y \oplus c$$

c.(3 points) Simplify and implement the single stage using NOR gates. Draw sketches.

2. (20 points) A 3-stage Gray Code synchronous Counter counts as follows (0,1,3,2,6,7,5,4).

a.(4 points) Draw the state diagram for the Counter.

b.(10 points) Design the 3-stage Counter using D-FF's and NAND gates.

c.(3 points) Test your implementation for the following two test inputs. Specifically, check for the next states for the following current states 3 and 4, i.e. 011 and 100.

d.(3 points) Include a reset control signal i.e. the Counter can be set to the '0' state by an external *control* input. Show only the state transition diagram for this.

3. (20 points) Design a circuit that reads in a positive 8 digit decimal number and converts its into binary equivalent. Each decimal digit is represented in 4 bit Binary coded decimal. The input number is read in one decimal digit at a time, starting from the least significant end. You may use the following components: full adders (these take two addend bits and a carry bit as inputs, and produce a carry bit and a sum bit as outputs), and nand gates. Keep the number of components used small.

4. (20 points) Design a circuit that will count the number of leading 0s in a N bit binary number. Assume that the number is supplied in parallel. You may use the following components: full adders (these take two addend bits and a carry bit as inputs, and produce a carry bit and a sum bit as outputs), and nand gates. Keep the number of components used small.

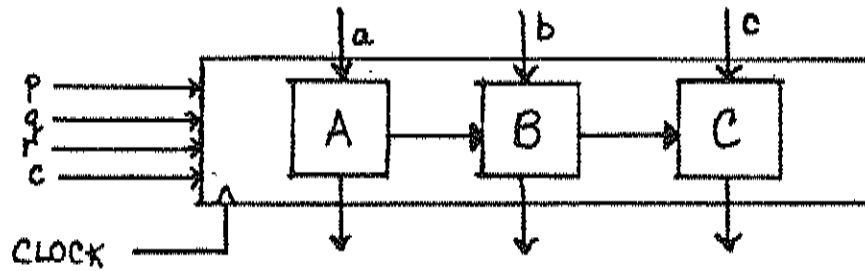
5. (20 points) Consider a memory containing N words that is used to store objects of varying sizes as follows:

- (a) The only allowable sizes are powers of 2, i.e. objects can be 1 word long, 2 words long, 4 words long and so on. The largest object could be N words long, where N is the size of the memory.
- (b) Each object is stored contiguously in memory.
- (c) Objects with size s are required to be aligned on an s word boundary, i.e. the starting address must be a multiple of s .

[A] (8 points) What is the lower bound on the number of bits required to identify objects stored in the memory? Note that the identifiers must all have the same length and must uniquely define the length of the object and its starting address in memory.

[B] (12 points) Design a scheme for constructing the identifiers. Your scheme should use as few bits as possible. Further, the starting address of the object and the length must be easy to calculate given the object identifier.

6. (20 points) A,B and C are three D Flip flops arranged as shown below.



p,q,r,c are control signals. The operation of the system is as follows:

Condition	Action
p=1	The contents of ABC are right shifted once.
q=1	The contents of ABC are left shifted once.
r=1	The contents of ABC are complemented.
c=1	The contents of ABC are set to zero

a.(10 points) Express the inputs to FF's as a function of Control inputs and the FF states.

b.(4 points) Implement the right shift function using NAND gates. Draw a sketch.

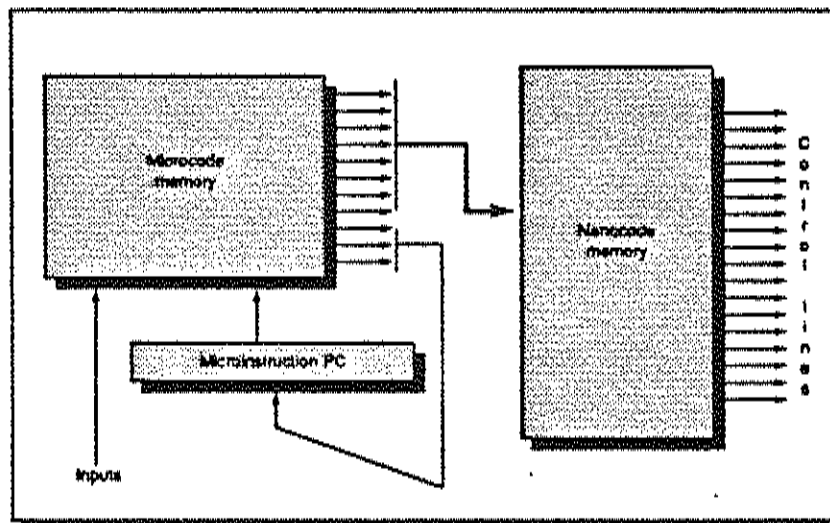
c.(3 points) Test your implementation of all functions for the following input.

$$ABC = 101$$

d.(3 points) What constraints do we have to place on the design and operation of the system? Be brief.

7. (20 points)

One technique that tries to get the best of both the worlds of vertical and horizontal microarchitectures is a *two-level* control store, as illustrated below. It tries to combine small control-store size with wide instructions. To avoid confusion, the bottom level uses the prefix *nano-*, yielding the terms 'nanoinstruction,' 'nanocode,' and so forth. This technique is used in the Burroughs D-machine. The idea is that the first level has many vertical instructions that point to the few unique horizontal instructions in the second level. The Burroughs D-machine was a general-purpose computer offering writable control store. Its microinstructions were 16 bits wide, with 12 of those bits specifying a nanoaddress, and the nanoinstructions were 56 bits wide. One instruction set interpreter used 1124 microinstructions and 123 nanoinstructions.



[A] (6 points) What is the general formula showing when a two-level control store scheme like Burroughs D-machine uses fewer bits than a single-level control store? Assume there are M microinstructions each a bits wide and N nanoinstructions each b bits wide.

[B] (4 points) Was the two-level control store of the D-machine successful in reducing control-store size versus a single-level control store for the interpreter?

[C] (4 points) After the code was optimized to improve cycles per instruction by 10 percent, the resulting code had 940 microinstructions and 161 nanoinstructions. Was the two-level control store of the D-machine successful in reducing control-store size versus a single-level control store for the optimized interpreter?

[D] (6 points) Did optimization increase or decrease the total number of bits needed to specify control? Why would the number of microinstructions decrease and the number of nanoinstructions increase?

8. (5 points) In a load-store machine the only operations that can access memory are loads and stores, while a memory-memory architecture can operate on operands directly in memory. Data collected from running programs on the two styles of machines revealed that 33 percent of the memory accesses were data references on the load-store machine while 70 percent of the memory accesses were for data on the memory-memory machine. What are two most likely technical reasons for this disparity?

9. (5 points) It is possible to use a unified cache for instructions and data, or alternatively, we could use separate caches for each. List the advantages and the disadvantages of each approach.

10. (40 points) In this problem you will give a high level design of a processor that will be able to execute vector operations. In particular the processor must be able to execute the following two instructions:

Add N, VA, VB, VC

This instruction adds two vectors each of length N, stored consecutively starting at address VA and address VB. The result is stored in N locations starting at address VC.

JGT LABEL

This causes a branch to address LABEL provided the result of the last addition performed was positive. If the result was non positive, the branch is not taken.

[A] (15 points) Give a block diagram implementation of such a device, using MSI components, e.g. registers, counters, latches, multiplexors, adders etc. You can assume that the adder completes a single addition each cycle, and that the memory can deliver a single word (data or instruction) each cycle. Use a straightforward approach; no need to be fancy and show any of the details, e.g. you may use just one block for a "control FSM", and you don't need to show any gate level details. Clearly list all the additional assumptions you make.

[B] (15 points) Give an RTL description for each instruction.

[C] (10 points) For the design above we assumed that memory could be accessed in a single cycle, whereas in reality, typical memory elements would be 5-10 times slower, so that each memory access would have a latency of 5-10 cycles. To overcome this problem, and to otherwise improve the performance of the machine which of the following feature or features would you incorporate? Briefly explain why, if at all, each feature would be useful, and whether you expect it to be cost effective.

*Answer: K. Burdige for
interleaved*

1. Multiport Memory.

2. Interleaved Memory.

3. Data Cache

4. Instruction Cache.

3

1. (10 points)

a. (4 points) Give the truth table for a single stage of a Full Binary Adder.

A	B	C _{in}	C	C _{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

(using $A + B + C_{in} = 2 \cdot C_{out} + C$)

4

b. (3 points) Express the Sum function and Carry Out function into standard sum of products form, and product of sums form.

5



	B			
A	0	1	0	1
	1	0	1	0
	C _{in}			

$$C = A B C_{in} \vee A \bar{B} \bar{C}_{in} \vee \bar{A} \bar{B} C_{in} \vee \bar{A} B \bar{C}_{in}$$

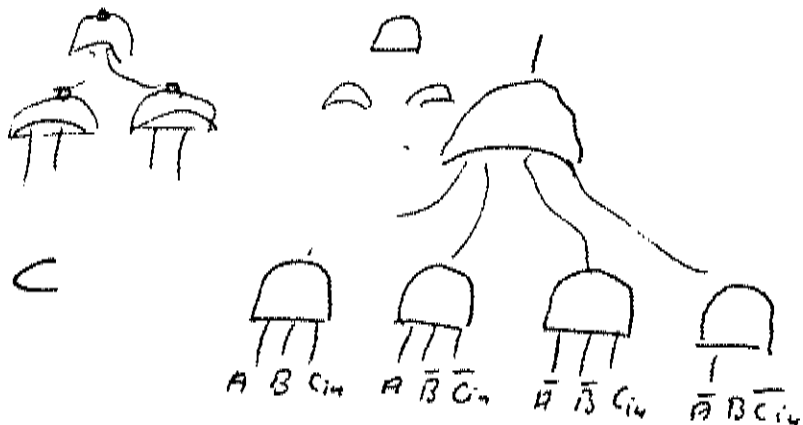
	B			
C _{out} A	1	0	1	0
	0	1	0	1
	C _{in}			

$$C_{out} = A B \vee A C_{in} \vee B C_{in}$$

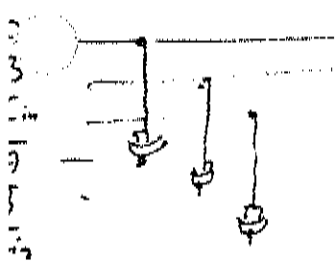
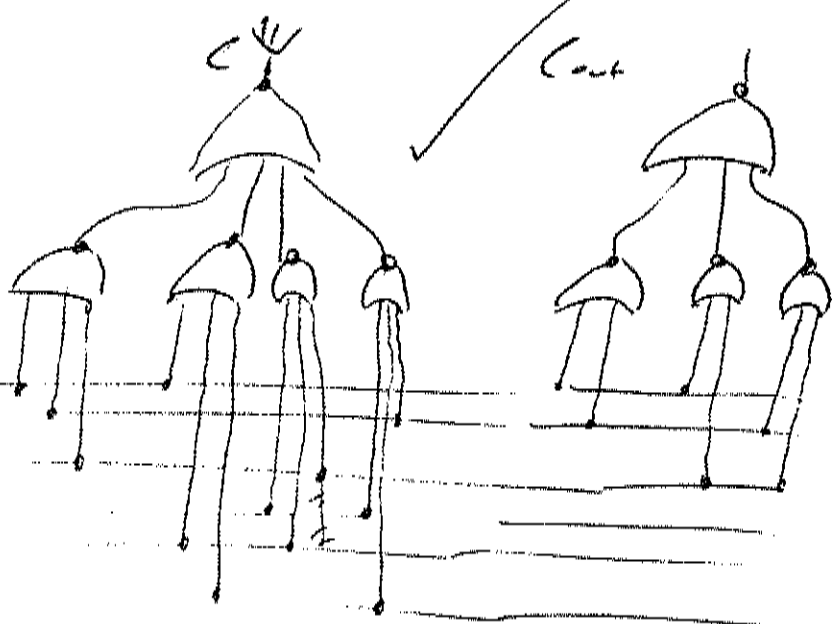
$$C = (\bar{A} \vee B \vee \bar{C}_{in}) \wedge (\bar{A} \vee \bar{B} \vee C_{in}) \wedge (A \vee \bar{B} \vee \bar{C}_{in}) \wedge (A \vee B \vee C_{in})$$

$$C_{out} = (A \vee C_{in}) \wedge (B \vee C_{in}) \wedge (A \vee B)$$

c. (3 points) Simplify and implement the single stage using NOR gates. Draw sketches.



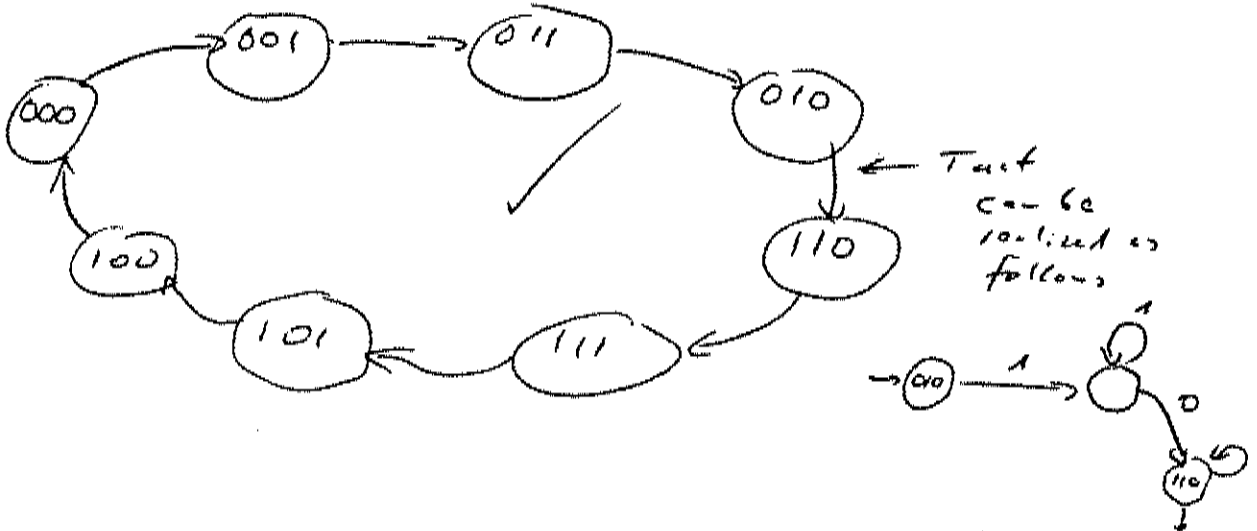
3



13

2. (20 points) A 3-stage Gray Code synchronous Counter counts as follows (0,1,3,2,6,7,5,4).

a. (4 points) Draw the state diagram for the Counter.



4

b. (10 points) Design the 3-stage Counter using D-FF's and NAND gates.

$= Q_1 = Q_2 = Q_3$

$S_0 S_1 S_2$	$S_0' S_1' S_2'$
000	001
001	011
011	010
010	110
110	111
111	101
101	100
100	000

equal to D inputs of D-FF's

S_0' Sol

	S_1	
S_2'	1 1 1 0	
	1 0 1 0	

S_1' Sol

	S_1	
S_2'	1 0 0 0	
	1 1 1 0	

$$S_0' = S_1 \bar{S}_2 + S_0 S_2$$

$$= \frac{S_1 \bar{S}_2 + S_0 S_2}{S_1 \bar{S}_2 + S_0 S_2}$$

$$S_1' = S_1 \bar{S}_2 + \bar{S}_0 S_2$$

$$= \frac{S_1 \bar{S}_2 + \bar{S}_0 S_2}{S_1 \bar{S}_2 + \bar{S}_0 S_2}$$

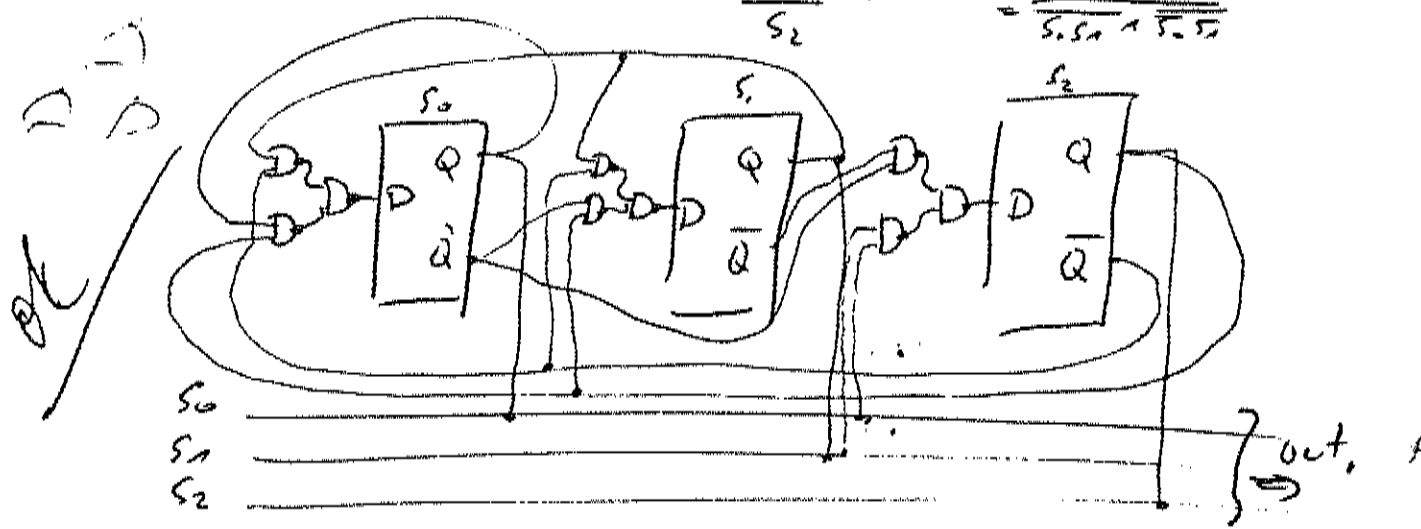
S_2' Sol

	S_1	
S_2'	1 1 0 0	
	0 0 1 0	

$$S_2' = S_0 S_1 + \bar{S}_0 \bar{S}_1$$

$$= \frac{S_0 S_1 + \bar{S}_0 \bar{S}_1}{S_0 S_1 + \bar{S}_0 \bar{S}_1}$$

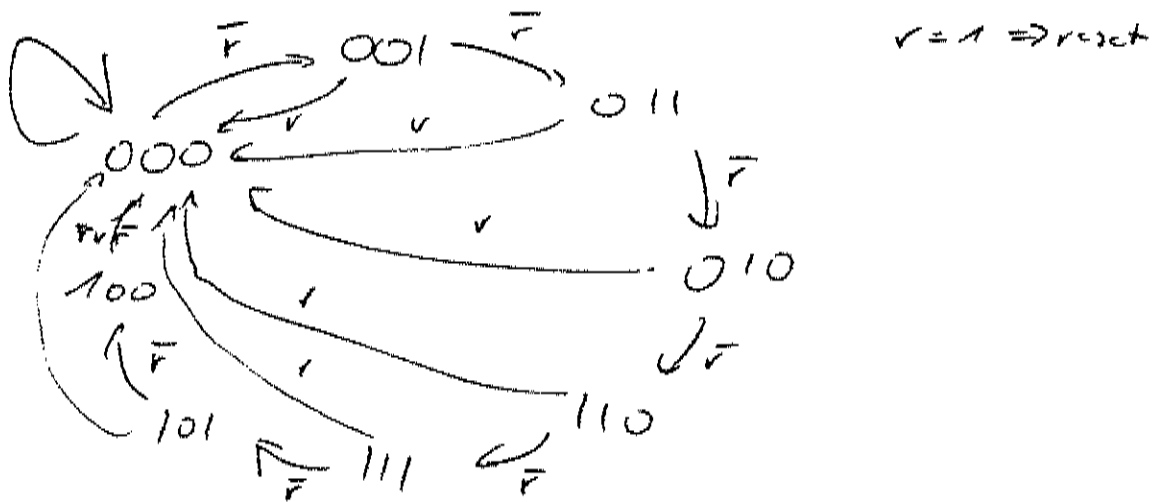
10



c. (3 points) Test your implementation for the following two test inputs. Specifically, check for the next states for the following current states 3 and 4, i.e. 011 and 100.

	s_0, s_1, s_2	s_0, s_1, s_2
	011	100
	↓	↓
s_0'	0	0
s_1'	1	0
s_2'	0	0
	⇒ 010 ✓	⇒ 000 ✓

d. (3 points) Include a reset control signal i.e. the Counter can be set to the '0' state by an external control input. Show only the state transition diagram for this.

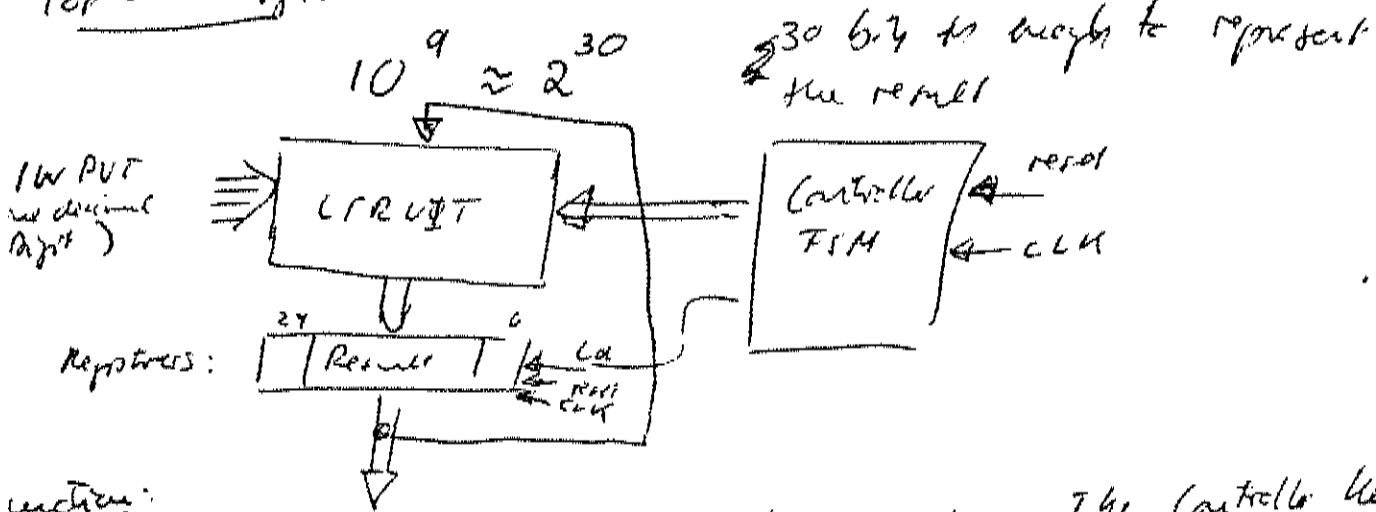


#20

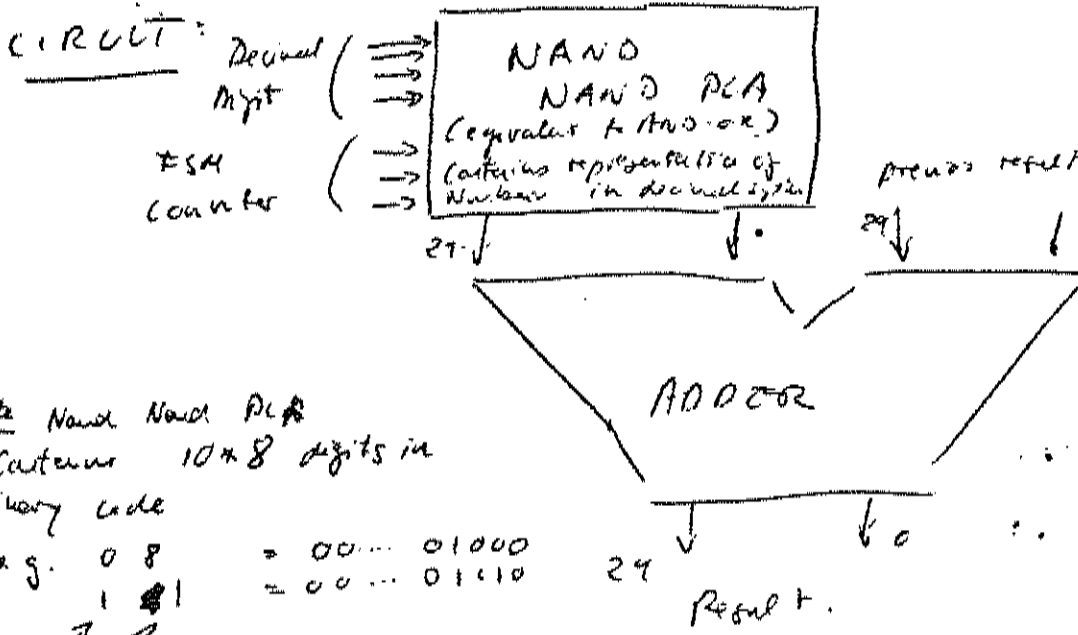
5

3. (20 points) Design a circuit that reads in a positive 8 digit decimal number and converts its into binary equivalent. Each decimal digit is represented in 4 bit Binary coded decimal. The input number is read in one decimal digit at a time, starting from the least significant end. You may use the following components: full adders (these take two addend bits and a carry bit as inputs, and produce a carry bit and a sum bit as outputs), and nand gates. Keep the number of components used small.

Top down design:



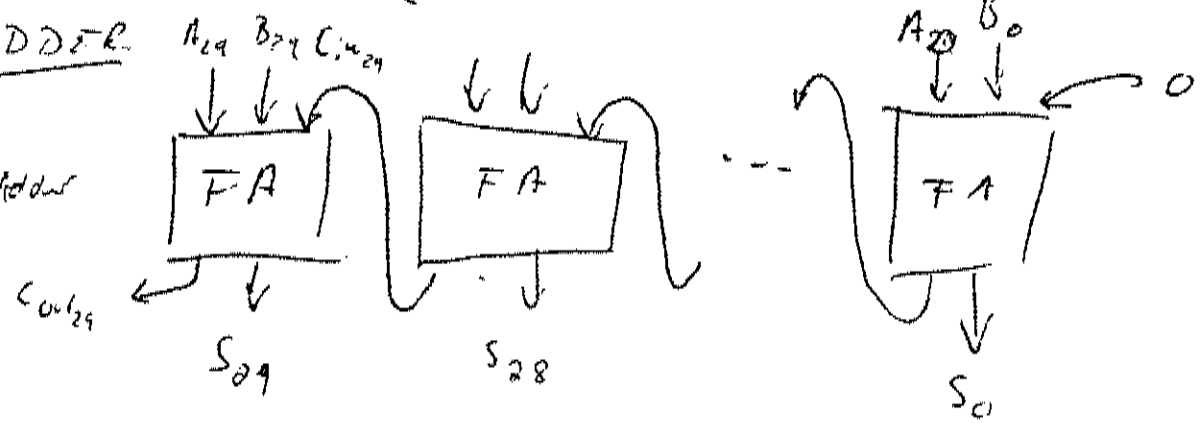
function: Every cycle one digit will be read in. The Controller keeps track of how many have already been read in. It uses a counter 0 to 7. If the least significant digit is read in first we start with 0. If the most significant digit comes first we start with 7 and count down. I assume the first case.



to NAND NAND PLA contains 10x8 digits in binary code
 e.g. 0 8 = 00...01000
 1 1 = 00...01110
 decimal digit

7 bit ADDER

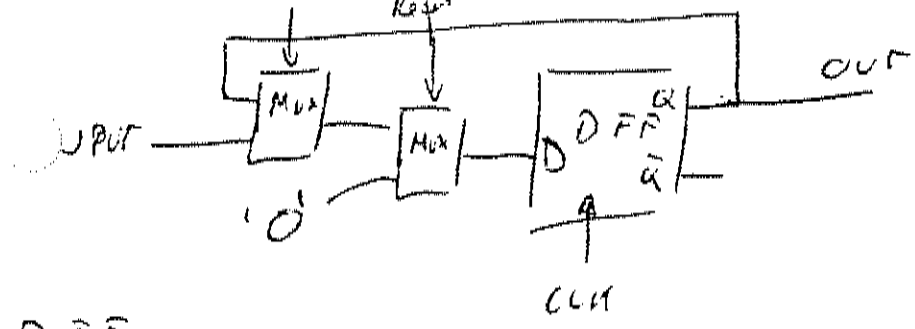
Full Adder



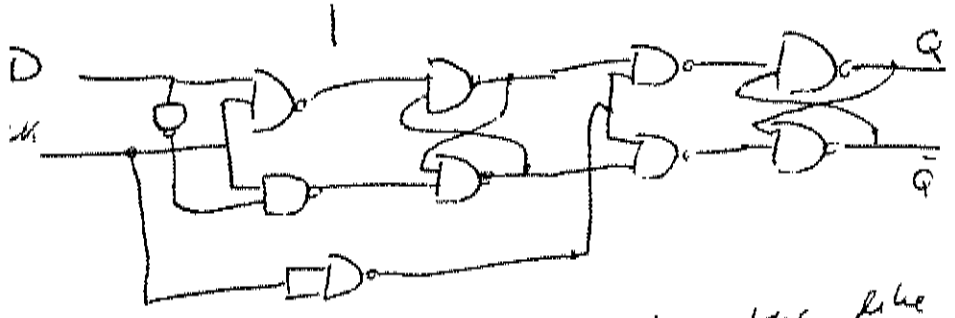
(ii) Register



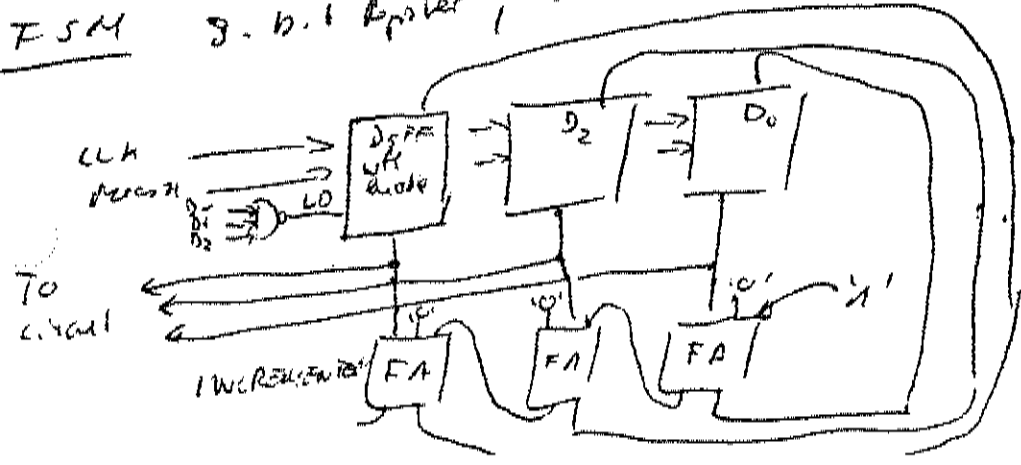
D-FF with Enable



D-FF



FSM 8-bit Register, 7 bit adder like above

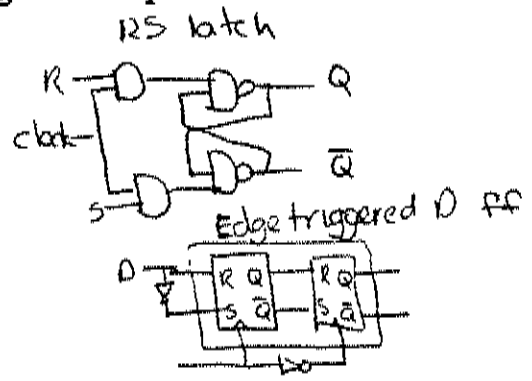
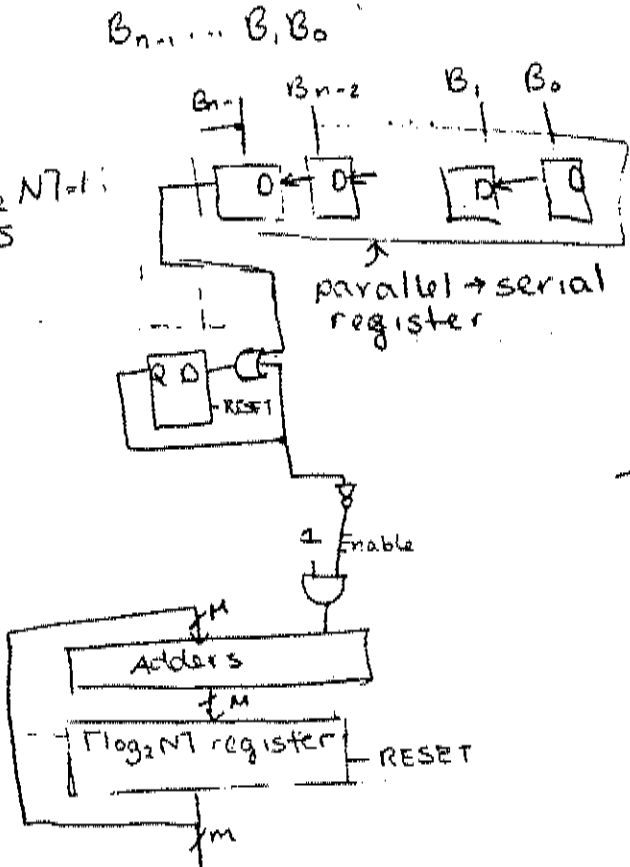


FSM
 If Reset is starts counting from 0 to 17 (111) and stops there. So Reset can be viewed as 'Start counting' for imp.

4. (20 points) Design a circuit that will count the number of leading 0s in a N bit binary number. Assume that the number is supplied in parallel. You may use the following components: full adders (these take two addend bits and a carry bit as inputs, and produce a carry bit and a sum bit as outputs), and nand gates. Keep the number of components used small.

20

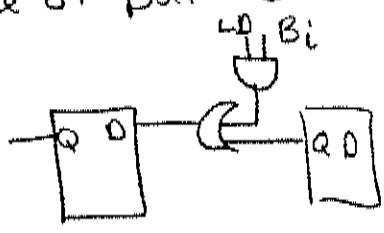
need $\lceil \log_2 N \rceil + 1$ to express answer



Transformed to serial to minimize components used.

This will add 1 to the register until a 1 is encountered. The Enable ff will remember the 1 and not allow the register to be incremented further

slice of par \rightarrow serial:



5. (20 points) Consider a memory containing N words that is used to store objects of varying sizes as follows:

- (a) The only allowable sizes are powers of 2, i.e. objects can be 1 word long, 2 words long, 4 words long and so on. The largest object could be N words long, where N is the size of the memory.
- (b) Each object is stored contiguously in memory.
- (c) Objects with size s are required to be aligned on an s word boundary, i.e. the starting address must be a multiple of s .

[A] (8 points) What is the lower bound on the number of bits required to identify objects stored in the memory? Note that the identifiers must all have the same length and must uniquely define the length of the object and its starting address in memory. (assume N is a power of 2)

need $\lceil \log_2(N-s) \rceil$ bits to address an object of size s . $\log_2 N$ if all same size

have $\log_2 N$ possible sizes

need $\lceil \log_2(\log_2 N) \rceil$ bits to identify size

so lower bound, given that all same size,

is $\log_2 N + \lceil \log_2(\log_2 N) \rceil$

Count # of ~~bits~~ address, length combinations.

[B] (12 points) Design a scheme for constructing the identifiers. Your scheme should use as few bits as possible. Further, the starting address of the object and the length must be easy to calculate given the object identifier.

Construct from address + length

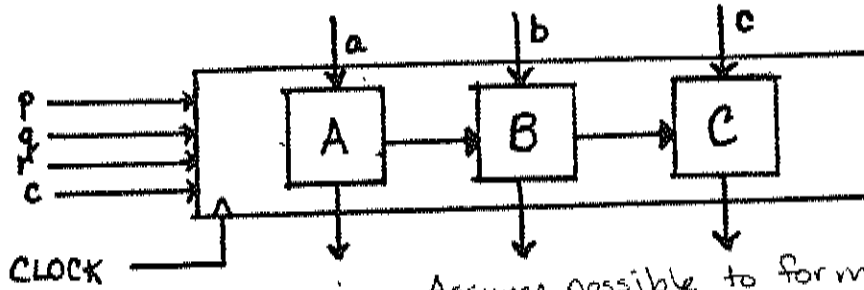
Concatenate address + \log_2 length

Take \log_2 by counting # of bits to the right of the leftmost 1 in the number.

Calculate ^{length} by putting a 1 in \log_2 length + 1 place from Identifier.

Since address is always same # of bits, easy to find where length starts.

6. (20 points) A, B and C are three D Flip flops arranged as shown below.



Assume possible to form connection beyond those shown (A → B, B → C), else impossible to left-shift

p, q, r, c are control signals. The operation of the system is as follows:

Condition	Action
p=1	The contents of ABC are right shifted once.
q=1	The contents of ABC are left shifted once.
r=1	The contents of ABC are complemented.
c=1	The contents of ABC are set to zero

} assume circular shift

a. (10 points) Express the inputs to FF's as a function of Control inputs and the FF states.

$$D_A = pQ_C + qQ_B + r\bar{Q}_A + \bar{p}\bar{q}\bar{r}cQ_A$$

$$D_B = pQ_A + qQ_C + r\bar{Q}_B + \bar{p}\bar{q}\bar{r}cQ_B$$

$$D_C = pQ_B + qQ_A + r\bar{Q}_C + \bar{p}\bar{q}\bar{r}cQ_C$$

Assume p, q, r, c run through priority encoder so that only 1 is high at a time. Also, if none are high, state remains unchanged. a, b, c are unused, since "contents of ABC" seems to mean current state of FF's.

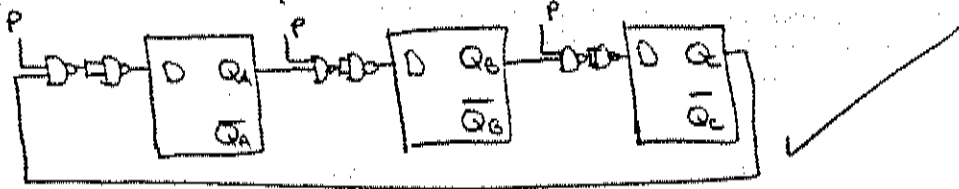
10

b.(4 points) Implement the right shift function using NAND gates. Draw a sketch.

$$D_A = pQ_c = \overline{pQ_c}$$

$$D_B = pQ_A = \overline{pQ_A}$$

$$D_C = pQ_B = \overline{pQ_B}$$



4

c.(3 points) Test your implementation of all functions for the following input.

$$ABC = 101$$

- $p=1 \quad D_A=1 \quad D_B=1 \quad D_C=0 \rightarrow 110 \checkmark$
- $q=1 \quad D_A=0 \quad D_B=1 \quad D_C=1 \rightarrow 011 \checkmark$
- $r=1 \quad D_A=0 \quad D_B=1 \quad D_C=0 \rightarrow 010 \checkmark$
- $c=1 \quad D_A=0 \quad D_B=0 \quad D_C=0 \rightarrow 000 \checkmark$

9

d.(3 points) What constraints do we have to place on the design and operation of the system? Be brief.

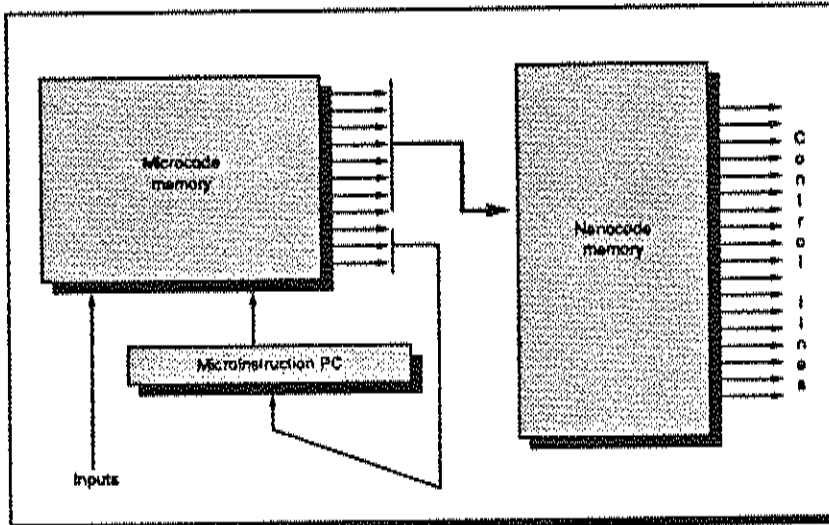
As stated already, it is important that only one of the control signals be active at any one time.

What about races, clock set up & hold times?

2

SPRING 1990 PRELIM QUESTIONS AND ANSWERS: 7 - 9

7 [20 points] One technique that tries to get the best of both the worlds of vertical and horizontal microarchitectures is a *two-level* control store, as illustrated by the figure below. It tries to combine small control-store size with wide instructions. To avoid confusion the bottom level uses the prefix *nano-*, yielding the terms "nanoinstruction," "nanocode," and so forth. This technique is in the Burroughs D-machine. The idea is that the first level has many vertical instructions that point to the few unique horizontal instructions in the second level. The Burroughs D-machine was a general-purpose computer offering writable control store. Its microinstructions were 16 bits wide, with 12 of those bits specifying a nanoaddress, and the nanoinstructions were 56 bits wide. One instruction set interpreter used 1124 microinstructions and 123 nanoinstructions.



a. [6] What is the general formula showing when a two-level control store scheme like Burroughs D-machine uses fewer bits than a single-level control store? Assume there are M microinstructions each a bits wide and N nanoinstructions each b bits wide.

Let t be the number of bits of a to specify the address of the nanoinstruction. t must be at least $\lceil \log_2 N \rceil$.

$$(M \times a + N \times b) < M \times (a - t + b)$$

or $(M \times t + N \times b) < M \times b$

or $N \times b < M \times (b - t)$

or $N < M \times (1 - t/b)$

or $t < (M - N)/M \times b$

[4 points for using $t = a$, unless say that assume does not include microinstruction PC bits]

[Get 1 points for 2 level portion correct, rest wrong]

7 cont'd

b. [4] Was the two-level control store of the D-machine successful in reducing control-store size versus a single-level control store for the interpreter?

$$\begin{aligned} & 1124 \times 16 + 123 \times 56 < 1124 \times (16 - 12 + 56) = 24872 < 67440 \\ \text{or} & 1124 \times 12 + 123 \times 56 < 1124 \times 56 = 20376 < 62944 \\ \text{or} & 123 \times 56 < 1124 \times (56-12) = 20376 < 62944 \end{aligned}$$

So the answer is yes; two-level did reduce control store size.

[-1 for using 1124×56 vs. 1124×60 in top formula. Even if got equation wrong, it was clear from explanation of hardware that single level needed more than 56 bits.]

[-0.5 for using $\log_2 123$ vs. 12. Problem states size of nanoinstruction address.]

[2 points if serious flaw in one piece]

c. [4] After the code was optimized to improve cycles per instruction by 10%, the resulting code had 940 microinstructions and 161 nanoinstructions. Was the two-level control store of the D-machine successful in reducing control-store size versus a single-level control store for the optimized interpreter?

$$\begin{aligned} & 940 \times 16 + 161 \times 56 < 940 \times (16 - 12 + 56) = 24056 < 56400 \\ \text{or} & 940 \times 12 + 161 \times 56 < 940 \times 56 = 20296 < 52640 \end{aligned}$$

So the answer is yes, once again two-level did reduce control store size.

[-1 point for using 940×56 vs. 940×60 in top formula. Even if got equation wrong, it was clear from explanation of hardware that single level needed more than 56 bits.]

[-0.5 for using $\log_2 161$ vs. 12. Problem states size of nanoinstruction address.]

d. [6] Did optimization increase or decrease the total number of bits needed to specify control? Why would the number of microinstructions decrease and the number of nanoinstructions increase?

[1] $24872 > 24056$, so optimization decreased the number of bits to specify control.

[5] Optimization tries to reduce execution time per instruction, usually by executing more operations in parallel. Executing more operations in parallel reduces the sharing of nanoinstructions since they are not as general, hence the increase in nanoinstructions. The number of microinstructions decreased because more operations per nanoinstruction means fewer microinstructions are needed to specify the operation.

[-2 for not explaining why fewer micro.]

8 [5] In a load-store machine the only operations that can access memory are loads and stores, while a memory-memory architecture can operate on operands directly in memory. Data collected from running programs on the two styles machines revealed that 33% of the memory accesses were data references on the load-store machine while 70% of the of the memory accesses were for data on the memory-memory machine. What are two most likely technical reasons for this disparity?

- Load-store computers executes more instructions than memory-memory architectures, thus a larger percentage of memory references are instructions. [3]
- Load-store computers keep data in registers and reuses data in registers, thereby having fewer memory accesses. [2]

9 [5] It is possible to use a unified cache for both instructions and data, or alternatively use, we could use separate caches for each. List the advantages and disadvantages of each approach.

Unified Advantages:

Better cache utilization since no rigid dividing line between instructions and data

If built from off-the-shelf parts, unified cache may be lowest cost since can supply both instructions and data from a single bank of RAM chips.

Simpler to implement: no check whether instruction or data before access goes to cache

No duplication: what if data and instructions aren't distinct.

Less time to build since only one cache

Split caches Advantages:

Twice the cache bandwidth: can fetch instructions and data simultaneously

No cache misses due to conflicts between instructions and data (or poor data locality won't lower instruction miss rates)

Dedicate cache policies depending on instructions vs. data (e.g., no writing, block size, cheaper data cache?)

One large cache might be slower than two smaller ones (memory access time)

[2 points per item, needing 3 items for full credit.]

[Usual problem with answer is listing an item without any indication why it was true. e.g., higher hit rate for split caches without mentioning that instructions and data can't conflict.]

University of California
College of Engineering
Department of Electrical Engineering
and Computer Sciences
Computer Science Division

HARDWARE PRELIM EXAM

Fall 1989

Your code number:

General Instructions:

- The exam last 3 hours = 180 minutes.
 - The exam has 27 pages.
 - Do all your work on these papers; use backsides if necessary.
 - Calculators are not allowed.
 - Read the problem statements carefully, at least twice.
 - You should not need to ask questions.
 - If something seems unclear, state your assumptions.
 - The indicated points give you an idea how long a problem should take (minutes).
 - There are 11 problems adding up to 200 points, so you have some choice.
 - Completing 180 points is considered a "perfect 10".
-

1

(15 min)

(a) Prove the following algebraically:

$$a + ab = a$$

$$ab + \bar{a}c + bc = ab + \bar{a}c$$

(b) Prove or disprove the following:

$$\bar{a}c + \bar{a}b + \bar{b}c + ab + a\bar{c} = a + b + c$$

$$ab + a\bar{c} + \bar{a}c = ac + bc + a\bar{c}$$

(c) Write in products of sums form:

$$\bar{a}b + \bar{c}(d + e)$$

(d) Use deMorgan's theorem to take the complement of:

$$a + bc$$

(e) Write the following in sum of products form:

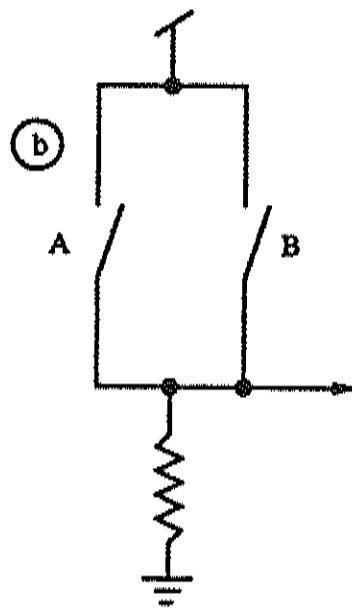
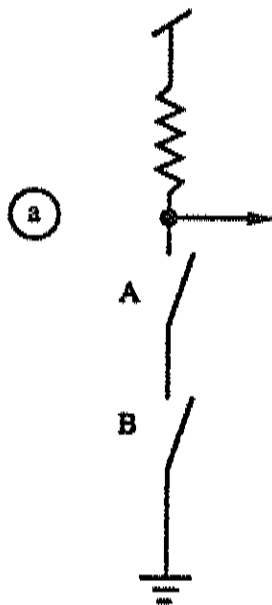
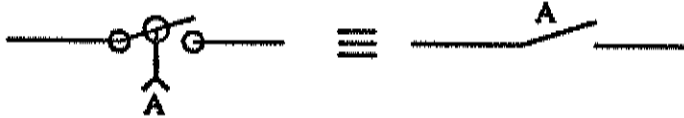
$$(a + \bar{b})(\bar{a} + cd)$$

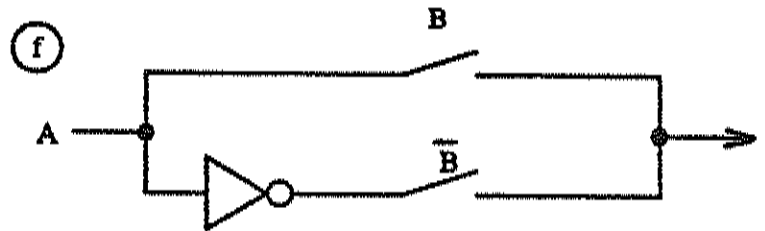
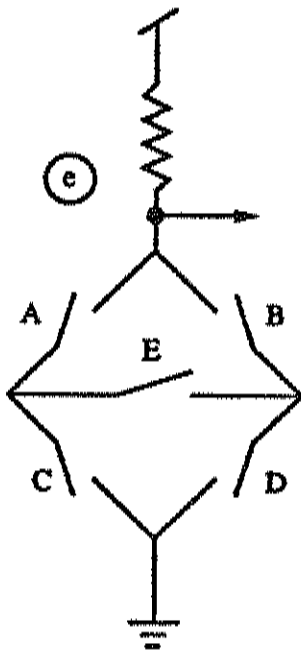
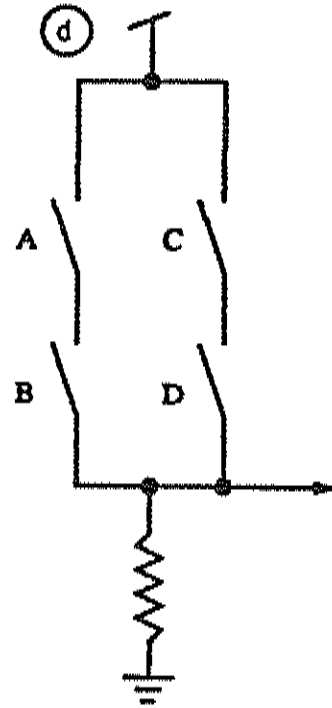
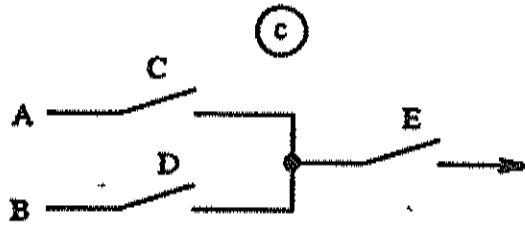
2

(10 min)

What is the function performed by each of the following?

note:





3

(10 min.)

Complete the following conversions:

(a) $(A1EF)_{16} = (?)_2$

(b) $(A1EF)_{16} = (?)_8$

(c) $(11101.110)_2 = (?)_{16}$

(d) $(11101.110)_2 = (?)_{10}$

(e) $(-23)_{10} = (?)$ in 8-bit signed 2's complement

(f) $(100)_{10} = (?)$ in 8-bit signed 2's complement

(g) (100110) in 2's complement = $(?)_{10}$

(h) Devise a gray code to represent the integers 0-7.

4

(15 min.)

Assume we have a computer with four I/O devices requesting service from the CPU. Each device is assigned one of four separate priorities ($P_0 - P_3$, P_3 the highest) and a corresponding request line. A device signals an interrupt to the CPU by asserting its request line. The CPU has only one interrupt input line, but has two inputs which specify an interrupt priority.

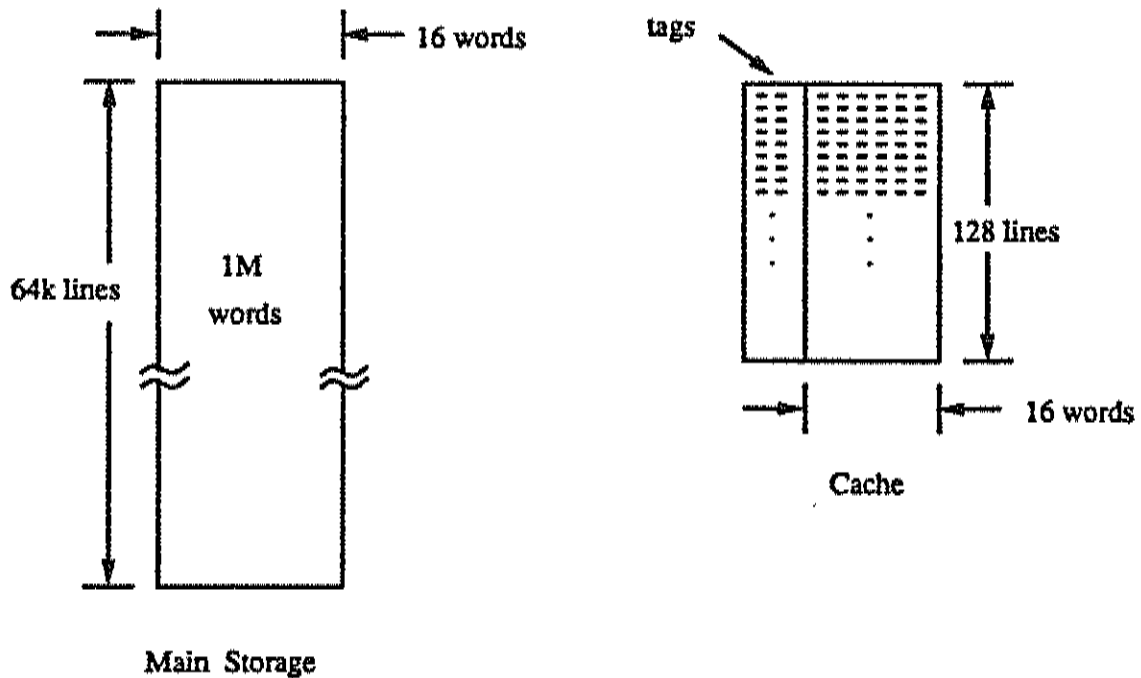
Design the circuit, called a priority encoder, which will interface the devices to the CPU by determining which line is requesting service at the highest priority and generating the proper CPU signals. Show the logic equations and gate implementation.

5

(20 min.)

Consider a word-addressable memory system with 1M (2^{20}) words of main storage and a cache consisting of 2048 words of data. The cache is organized as 128 lines (blocks) with each line holding 16 words of data and one tag. The CPU references the memory system with 20-bit addresses.

- For
- a) a fully-associative cache,
 - b) a direct-mapped cache,
 - c) an eight-way set associative cache,
- (i) how many tag bits per line must be stored with each line in the cache?
 - (ii) partition the address from the CPU into fields and state how each field is to be used to access a word from the memory system.



(a)

[Faint, illegible text]

(b)

[Faint, illegible text]

(c)

[Faint, illegible text]

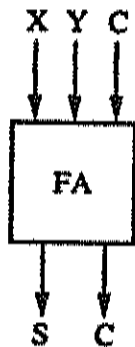
6

(30 min.)

- (a) From the following set of parts you are to design several versions of a circuit to multiply two n -bit positive integers, A and B . Version 1 should multiply the numbers using the minimal amount of time, and version 2 using the minimal number of full-adder (FA) cells.

A and B are initially in two n -bit registers and the $2n$ bit result should end up in a register. You may assume the presence of clocks and control signals, but clearly explain in words the function of your control signals.

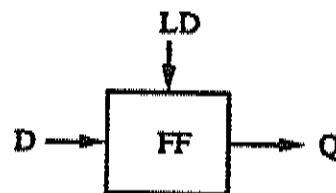
Keep it simple!



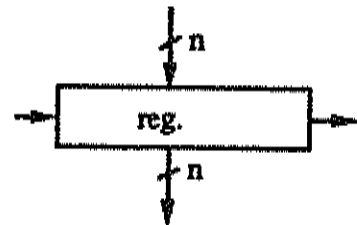
full-adder



and-gate



D-flip flop

parallel-load
shift register

Version 1:

Version 2:

- (b) As we increase n , how does the size and speed of both versions of the multiplier scale?

7

(35 min.)

You are given the pipelined arithmetic processor shown on the next page. It consists of a floating point multiplier, a floating point adder, with an input Bus 'A' and one output bus 'Y'. Operands are stored in a DATA RAM which is not dual ported.

An address unit is used to select operands. It consists of an address register file with four index registers, R0—R3, which can be incremented or decremented.

Timing: a register load operation, and memory read and write take 1 clock cycle. Floating point operations take 10 clock cycles.

In this problem you will use Register Transfer Language (RTL) to describe how macro instructions would be implemented in horizontal micro code.

Example: Macro instruction SQUARE (R0); multiply contents of RAM at location R0 by itself.

Micro Instructions:

R0 → MAR; get operand address

MDO → MUL0, MDO → MUL1; write operand simultaneously into multiplier multiplicand reg.

NOP

NOP

NOP

NOP

; wait for multiply to complete

NOP

NOP

NOP

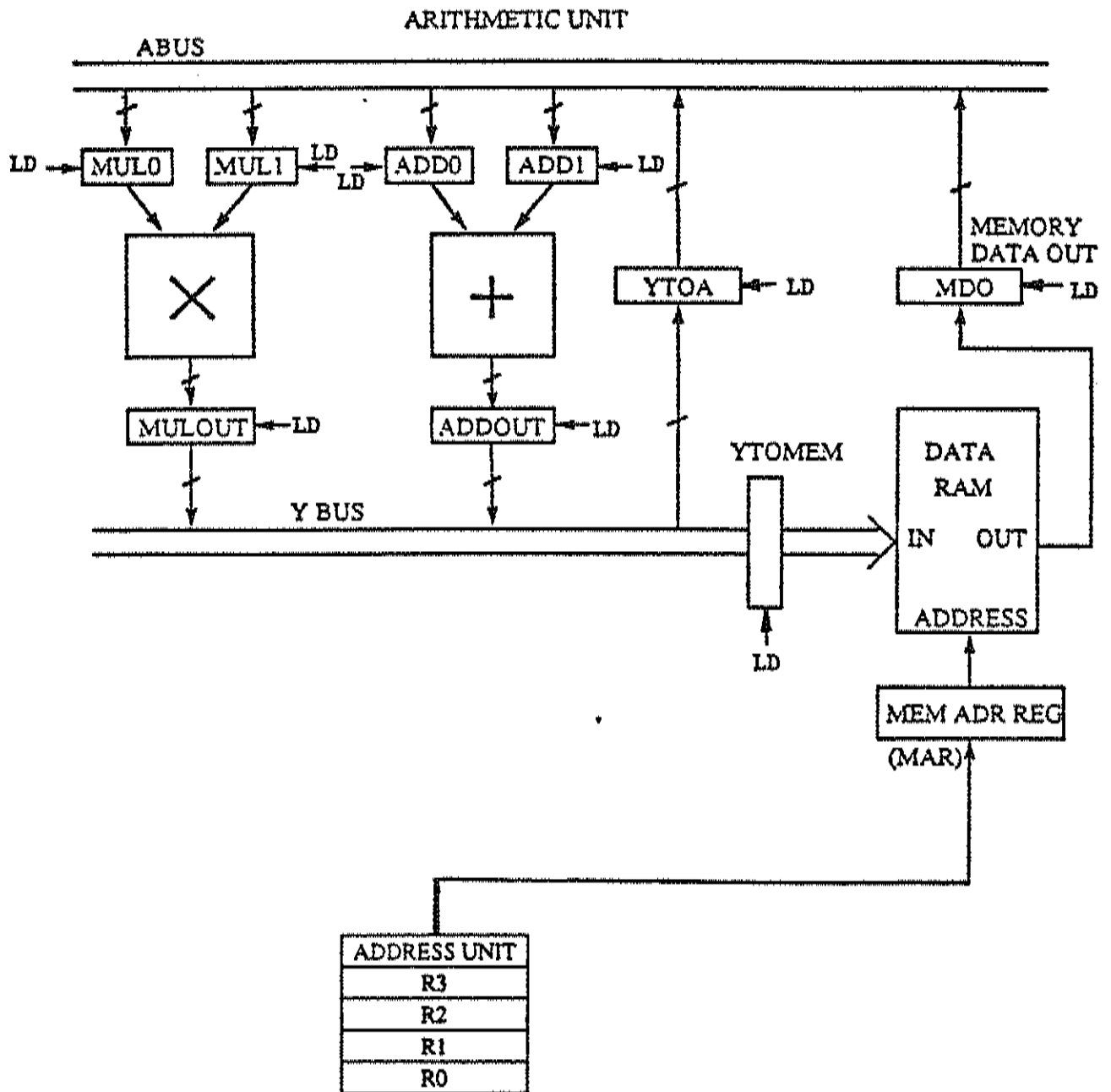
NOP

NOP

NOP

MULOUT → YTOMEM; write into (R0).

Assume Horizontal Microcode -- Assume all data regs are initially 0.



- (A) Give RTL description for the instruction
ADD (R0),(R1),(R2) which performs $(R0) + (R1) \rightarrow (R2)$
The two operands are pointed to by R0 and R1 respectively, and
the result is stored in the location pointed to be R2. Include com-
ments as appropriate. Use the answer sheet on the next page.
- (B) Give RTL description for the "dot" product of two vectors A and B
 $\vec{A} = (a_1, a_2, a_3, \dots, a_n)$, $\vec{B} = (b_1, b_2, b_3, \dots, b_n)$. $\vec{A} \cdot \vec{B} = a_1b_1 + a_2b_2 +$
 $a_3b_3 + \dots a_nb_n$.

R0 points to vector A. R2 is location of result,
R1 points to vector B. R3 is the length of the vector.

Maximize throughput by keeping all adders and multipliers busy.

A)

Program Step	Operation(s)	Comments
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		
17		
18		
19		
20		

B)

Program Step	Operation(s)	Comments
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		
17		
18		
19		
20		

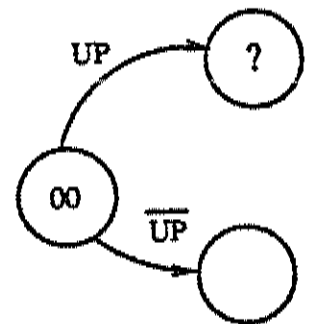
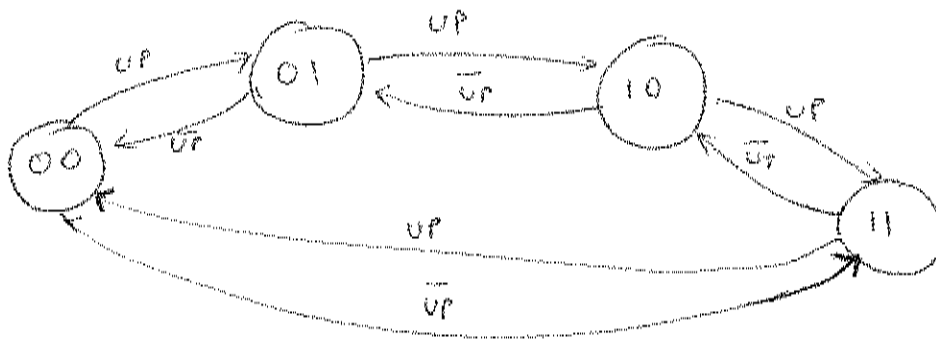
8

(15 min.)

In this problem you will design a synchronous 2 bit up/down binary counter using rising edge triggered JK Type Flip Flops. The output of the counter must be glitch free. The block diagram of the counter is:



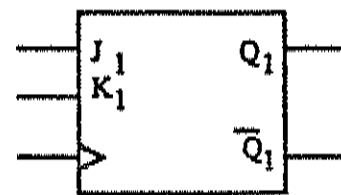
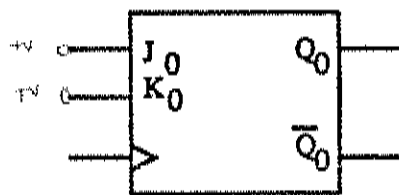
(A) draw a state diagram for the counter, using notation shown to the right as an example.



(B) Fill in the next state table for the up/down counter.

PRESENT STATE		INPUT	CONTROLS				NEXT STATE	
Q_1	Q_0	Up/Down	J_0	K_0	J_1	K_1	Q_1	Q_0
0	0	1	1	x	0	x	0	1
0	0	0	1	x	1	x	1	1
0	1	1	x	1	1	x	1	0
0	1	0	x	1	0	x	0	0
1	0	1	1	x	x	0	1	1
1	0	0	1	x	1 0	1 0	0	1
1	1	1	x	1	x 0	x 1	0	0
1	1	0	x	1	x	0	1	0

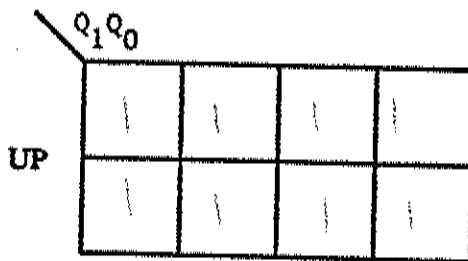
8(Cont.) Complete the schematic of the counter, using minimized logic.



k-maps (for your convenience)

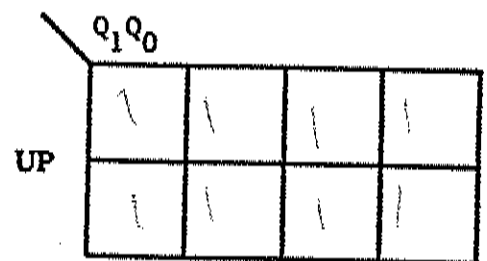
$J_0:$

$J_0 = 1$



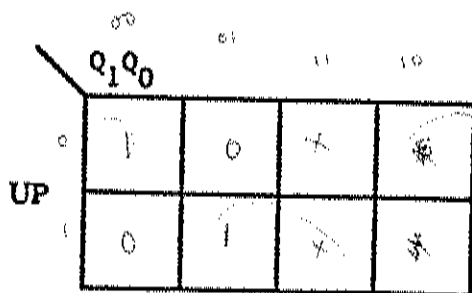
$K_0:$

$K_0 = 1$



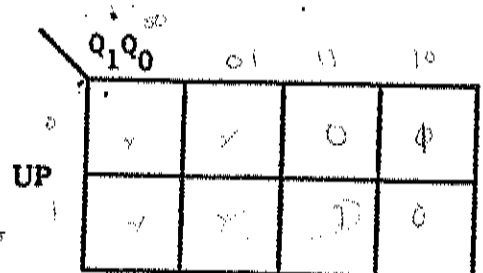
$J_1:$

$J_1 = \overline{Q_1} Q_0 + \overline{Q_1} \overline{Q_0}$



$K_1:$

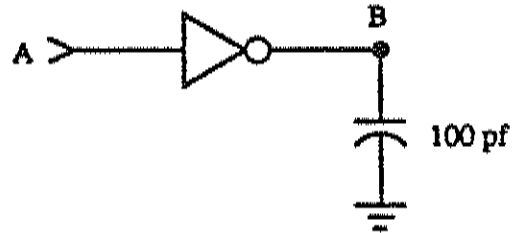
$K_1 = \overline{Q_1} Q_0 + \overline{Q_1} \overline{Q_0}$



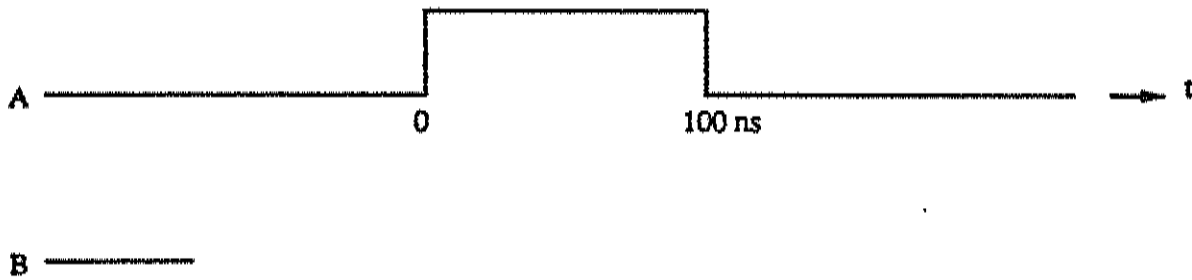
9

(5 min.)

Consider a logic device with 100Ω source impedance driving a 100pF capacitive load. With no load, the driver switches between 0.0 and 5.0 volts DC.



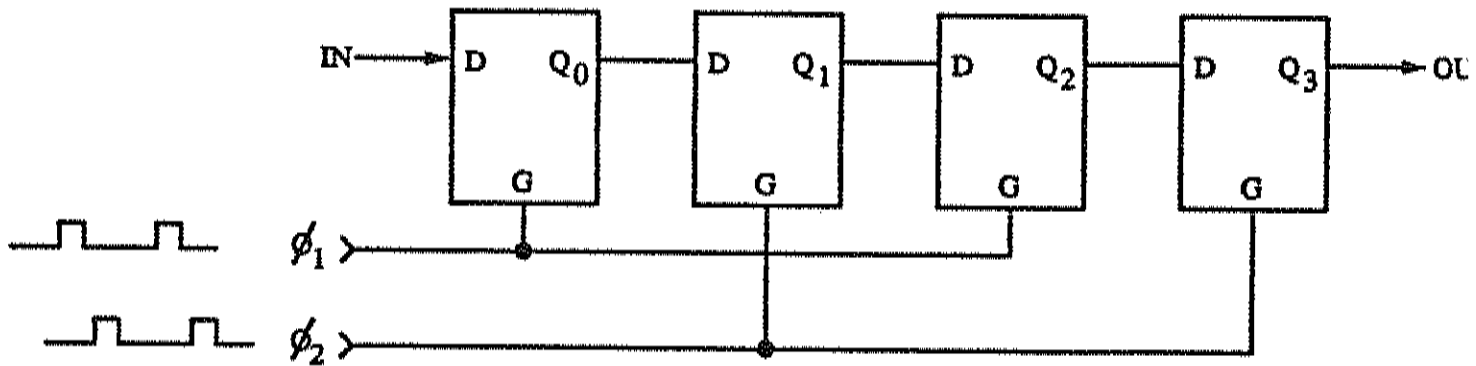
Neglecting internal propagation delay through the inverter, complete the following sketch.



10

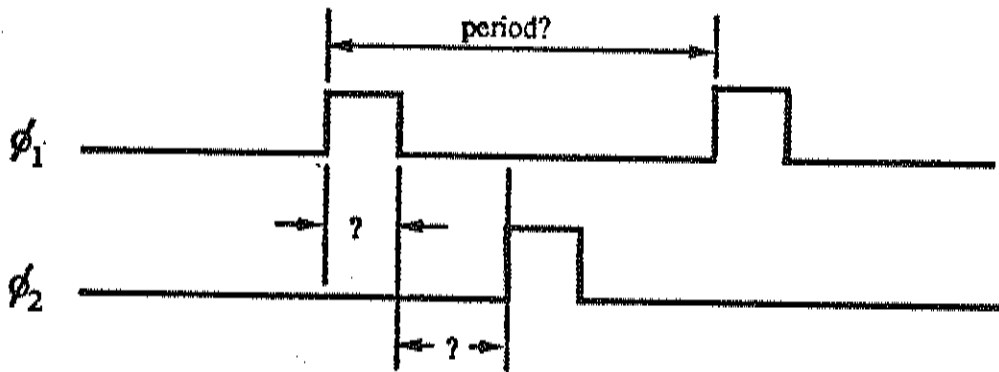
(15 min.)

Below is a 4 bit shift register using a two phase clock, and level-sensitive (not edge triggered) type D latches. The two clock phases are strictly non-overlapping ideal rectangular clocks. The latches have a 20 ns maximum delay from data-in to data-out. (Hold time from when the clock goes high is also 20 ns).



- (A) With no clock skew, what is the maximum frequency of operation? Give a simple timing diagram to justify your answer.

(B) Specify clock parameters for part A.



(C) If ϕ_2 can be skewed (-0 ns, $+30$ ns) with respect to ϕ_1 , what is the maximum frequency of operation? Give a simple timing diagram to justify your answer.

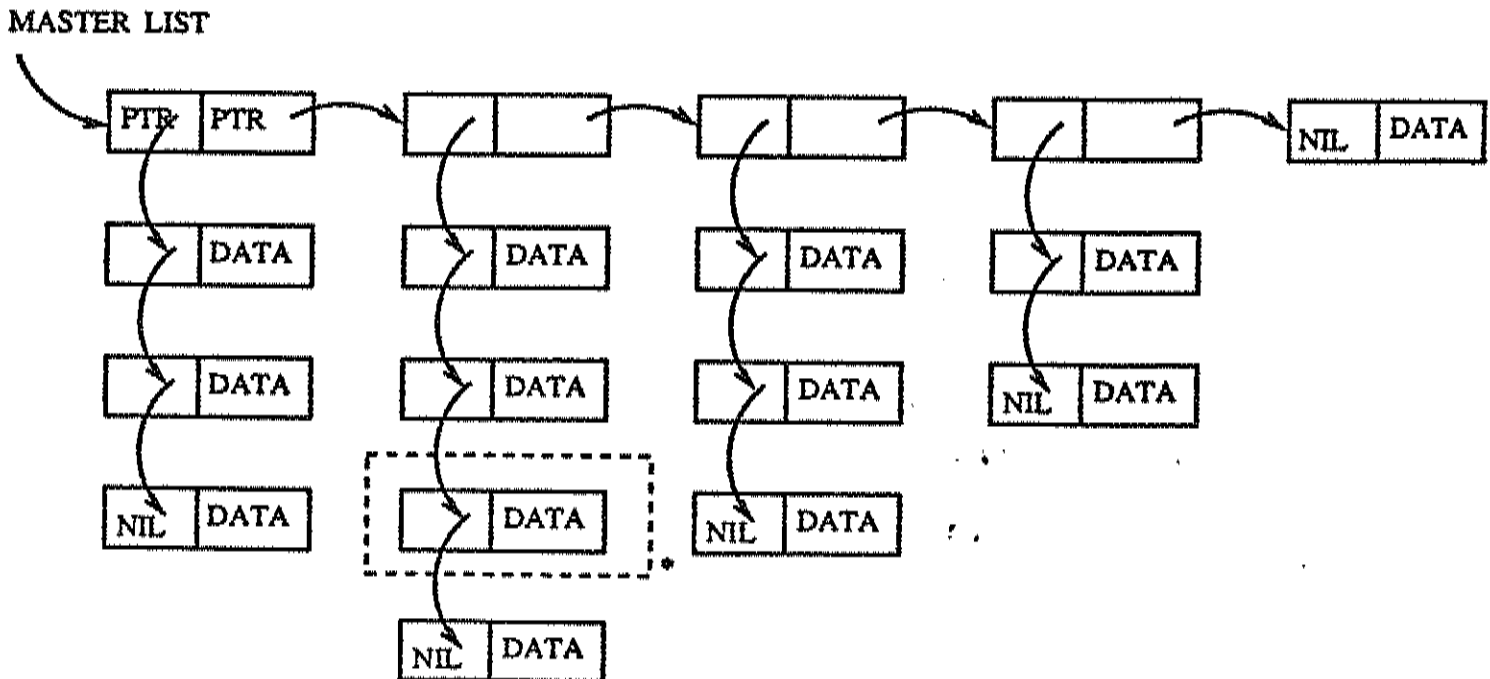
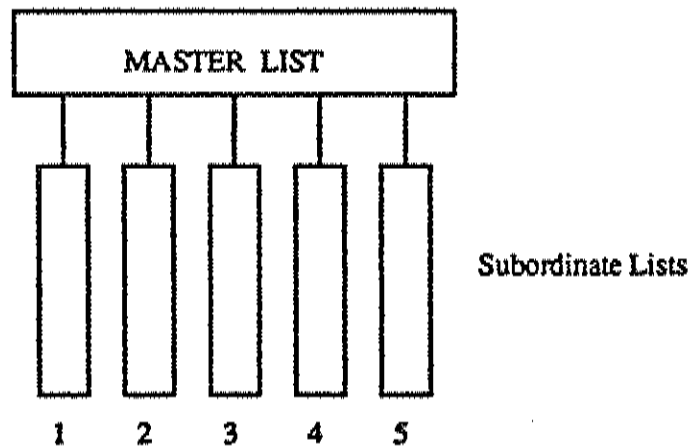
11

(30 min.)

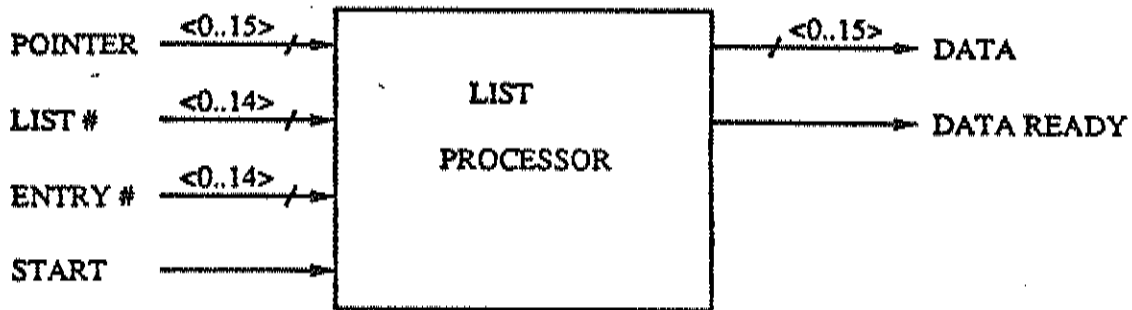
In this problem, you will be doing the high level design for a list processor. The design is built around a 64K x 16 memory which contains a "list of lists". The internal organization is shown below:

A cell is composed of two 16 bit words. For the master list, the first element in the cell points to a sublist, the second element points to the next cell in the master list. For the sublist, the first element in the cell is a pointer to the next cell, and the second element is a 16 bit word of data. A special pointer value of 'NIL' is used to indicate the last entry in a list.

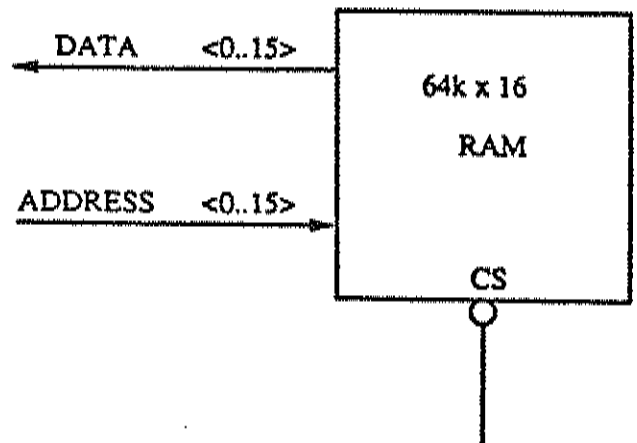
The third entry of the second list is marked with an asterisk.



Operation of list processor:



Pointer is address of root of list. LIST and ENTRY specify which entry of which sublist to return. On START assertion, the List Processor begins to search down the master and sublist. Once the appropriate data is found it is sent out on DATA $\langle 0..15 \rangle$ and DATA-READY is asserted. Assume that a NIL will not be encountered before the desired entry is found.



- (A) Give a block diagram implementation of such a device, using MSI components: registers, counters, flip flops, multiplexors, adders, comparators, etc., and a 64K x 16 RAM. Choose a straight-forward approach; no need to be fancy or show any details, e.g., you may just show one block for a "control FSM", and you don't need to show any gate level details.

(B) SPECIFY operation of the list processor using RTL.

University of California
College of Engineering
Department of Electrical Engineering
and Computer Sciences
Computer Science Division

HARDWARE PRELIM EXAM

Fall 1989

Your code number:

Key

General Instructions:

- The exam last 3 hours = 180 minutes.
 - The exam has 27 pages.
 - Do all your work on these papers; use backsides if necessary.
 - Calculators are not allowed.
 - Read the problem statements carefully, at least twice.
 - You should not need to ask questions.
 - If something seems unclear, state your assumptions.
 - The indicated points give you an idea how long a problem should take (minutes).
 - There are 11 problems adding up to 200 points, so you have some choice.
 - Completing 180 points is considered a "perfect 10".
-

1

(15 min)

(a) Prove the following algebraically:

$$\begin{aligned}
 a + ab &= a \\
 a(1 + b) & \\
 a(1) & \\
 a &= a
 \end{aligned}$$

$$ab + \bar{a}c + bc = ab + \bar{a}c$$

$$\begin{aligned}
 ab + \bar{a}c + (a + \bar{a})bc & \\
 a(b + bc) + \bar{a}(c + bc) & \\
 ab + \bar{a}c &
 \end{aligned}$$

(b) Prove or disprove the following:

$$\bar{a}c + \bar{a}b + \bar{b}c + ab + a\bar{c} = a + b + c$$

Equal

a	b	c	LHS	RHS
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

$$ab + a\bar{c} + \bar{a}c = ac + bc + a\bar{c}$$

Not Equal

a	b	c	LHS	RHS
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	1	1
1	0	0	1	1
1	0	1	0	1
1	1	0	1	1
1	1	1	1	1

(c) Write in products of sums form:

$$f = \bar{a}b + \bar{c}(d+e) = \bar{a}b + \bar{c}d + \bar{c}e$$

from k-map or truth table

$$\bar{f} = ac + \bar{a}\bar{b}c + \bar{b}\bar{d}\bar{e} + a\bar{d}\bar{e}$$

by deMorgan's:

$$\begin{aligned} f = \bar{\bar{f}} &= \overline{ac + \bar{a}\bar{b}c + \bar{b}\bar{d}\bar{e} + a\bar{d}\bar{e}} \\ &= (\bar{a} + \bar{c})(b + \bar{c})(\bar{a} + d + e)(b + e + d) \end{aligned}$$

(d) Use deMorgan's theorem to take the complement of:

$$\overline{a+bc} = \bar{a}(\overline{bc}) = \bar{a}(\bar{b} + \bar{c})$$

(e) Write the following in sum of products form:

$$(a + \bar{b})(\bar{a} + cd) =$$

$$a\bar{a} + acd + \bar{a}\bar{b} + \bar{b}cd =$$

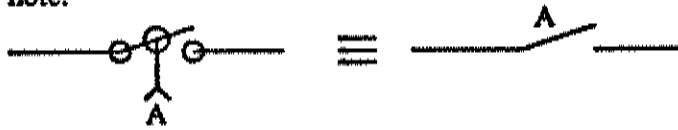
$$acd + \bar{a}\bar{b} + \bar{b}cd$$

2

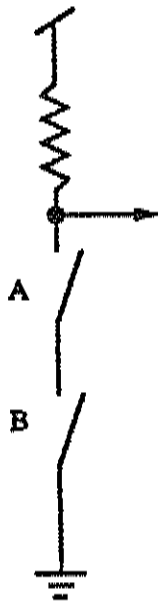
(10 min)

What is the function performed by each of the following?

note:

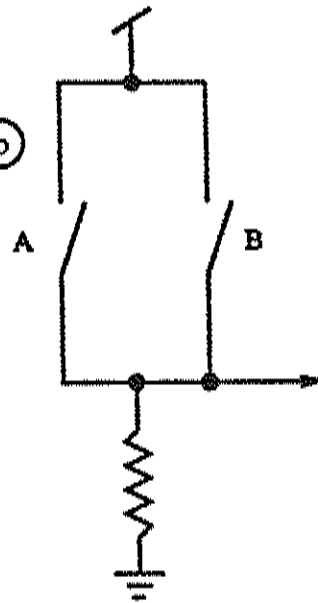


a

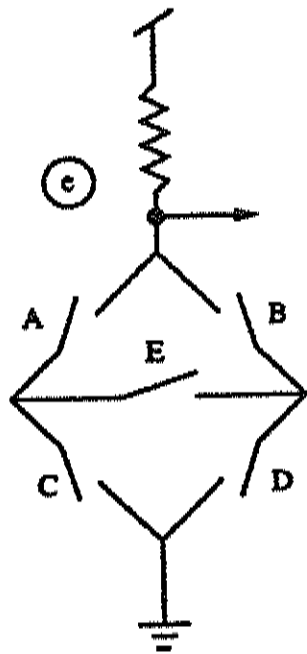
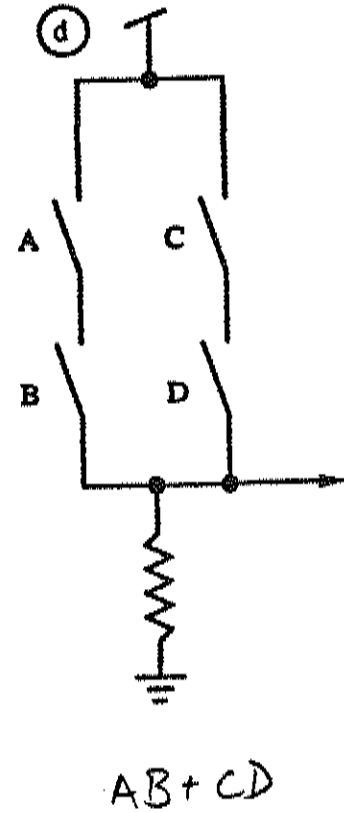
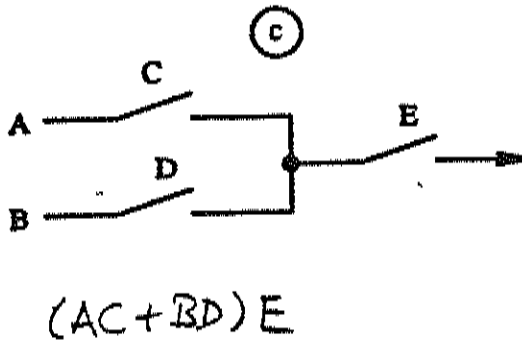


NAND(A,B)

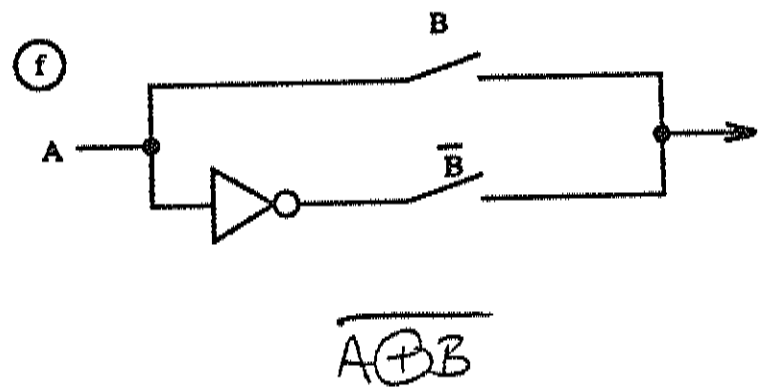
b



OR(A,B)



$\overline{AC + BD + AED + BEC}$



3

(10 min.)

Complete the following conversions:

$$(a) (A1EF)_{16} = (?)_2 = (1010\ 0001\ 1110\ 1111)_2$$

$$(b) (A1EF)_{16} = (?)_8 = (120757)_8$$

$$(c) (11101.110)_2 = (?)_{16} = (1D.C)_{16}$$

$$(d) (11101.110)_2 = (?)_{10} = (29.75)_{10}$$

$$(e) (-23)_{10} = (?) \text{ in 8-bit signed 2's complement} = 1110\ 1001$$

$$(f) (100)_{10} = (?) \text{ in 8-bit signed 2's complement} = 0110\ 0100$$

$$(g) (100110) \text{ in 2's complement} = (?)_{10} = -26$$

(h) Devise a gray code to represent the integers 0-7.

0	1	2	3	4	5	6	7
000	001	011	111	101	100	110	010

4

(15 min.)

Assume we have a computer with four I/O devices requesting service from the CPU. Each device is assigned one of four separate priorities ($P_0 - P_3$, P_3 the highest) and a corresponding request line. A device signals an interrupt to the CPU by asserting its request line. The CPU has only one interrupt input line, but has two inputs which specify an interrupt priority.

Design the circuit, called a priority encoder, which will interface the devices to the CPU by determining which line is requesting service at the highest priority and generating the proper CPU signals. Show the logic equations and gate implementation.

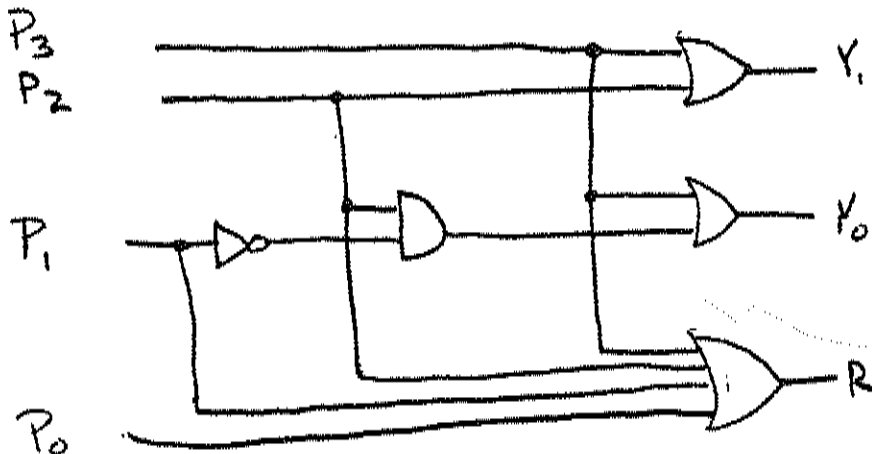
Encode (Y_1, Y_0) from (00) to (11) to represent the requests on lines P_2 to P_3 respectively. Line R will interrupt the CPU.

P_3	P_2	P_1	P_0	Y_1	Y_0	R
0	0	0	0	-	-	0
1	-	-	-	1	1	1
0	1	-	-	1	0	1
0	0	1	-	0	1	1
0	0	0	1	0	0	1

$$Y_1 = P_3 + P_2$$

$$Y_0 = P_3 + \bar{P}_2 P_1$$

$$R = P_3 + P_2 + P_1 + P_0$$



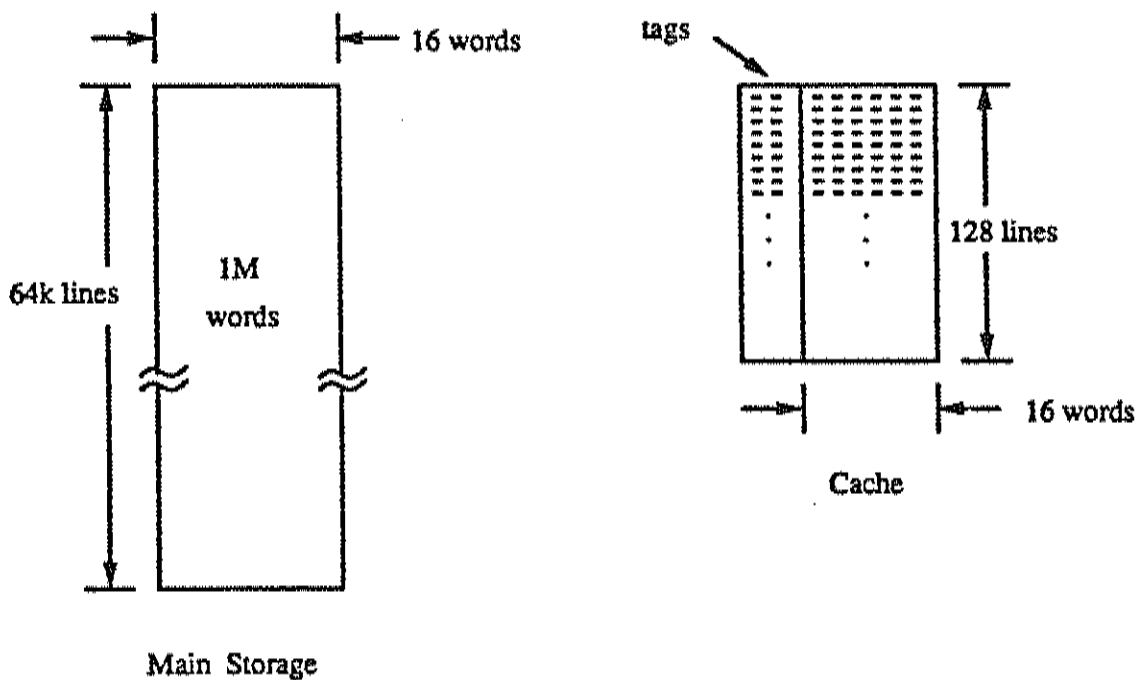
5

(20 min.)

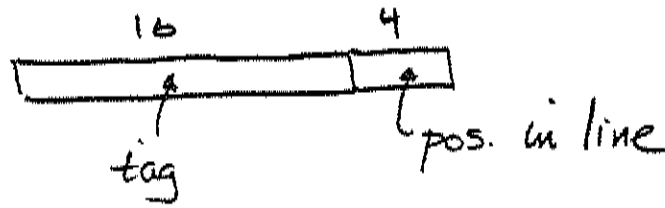
Consider a word-addressable memory system with 1M (2^{20}) words of main storage and a cache consisting of 2048 words of data. The cache is organized as 128 lines (blocks) with each line holding 16 words of data and one tag. The CPU references the memory system with 20-bit addresses.

- For
- a) a fully-associative cache,
 - b) a direct-mapped cache,
 - c) an eight-way set associative cache,

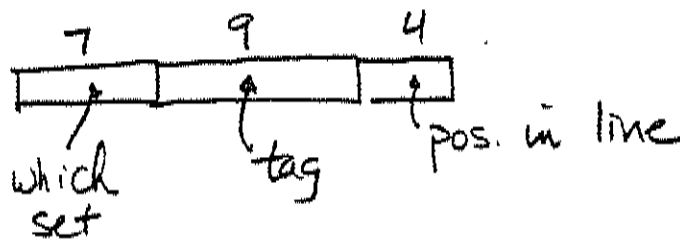
- (i) how many tag bits per line must be stored with each line in the cache?
- (ii) partition the address from the CPU into fields and state how each field is to be used to access a word from the memory system.



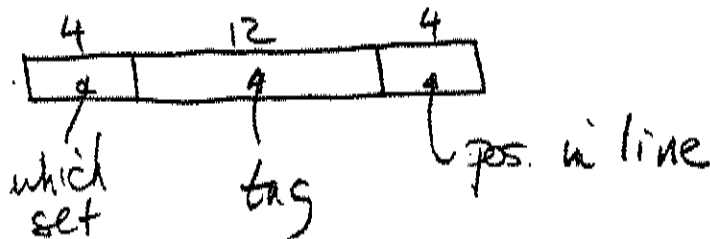
(a) Fully-associative 16 tag bits



(b) direct-mapped 9 tag-bits



(c) eight-way set associative 12 tag bits



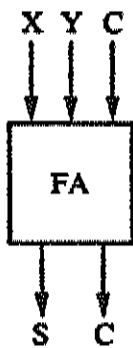
6

(30 min.)

- (a) From the following set of parts you are to design several versions of a circuit to multiply two n -bit positive integers, A and B . Version 1 should multiply the numbers using the minimal amount of time, and version 2 using the minimal number of full-adder (FA) cells.

A and B are initially in two n -bit registers and the $2-n$ bit result should end up in a register. You may assume the presence of clocks and control signals, but clearly explain in words the function of your control signals.

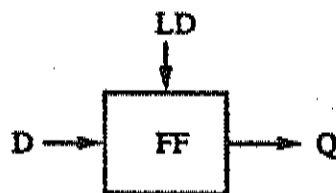
Keep it simple!



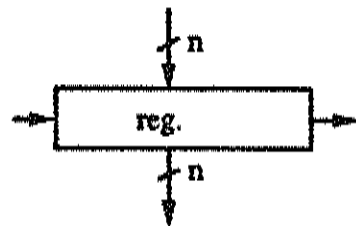
full-adder



and-gate



D-flip flop

parallel-load
shift register

7

(35 min.)

You are given the pipelined arithmetic processor shown on the next page. It consists of a floating point multiplier, a floating point adder, with an input Bus 'A' and one output bus 'Y'. Operands are stored in a DATA RAM which is not dual ported.

An address unit is used to select operands. It consists of an address register file with four index registers, R0—R3, which can be incremented or decremented.

Timing: a register load operation, and memory read and write take 1 clock cycle. Floating point operations take 10 clock cycles.

In this problem you will use Register Transfer Language (RTL) to describe how macro instructions would be implemented in horizontal micro code.

Example: Macro instruction SQUARE (R0); multiply contents of RAM at location R0 by itself.

Micro Instructions:

R0 → MAR; get operand address

MDO → MUL0, MDO → MUL1; write operand simultaneously into multiplier multiplicand reg.

NOP

NOP

NOP

NOP

; wait for multiply to complete

NOP

NOP

NOP

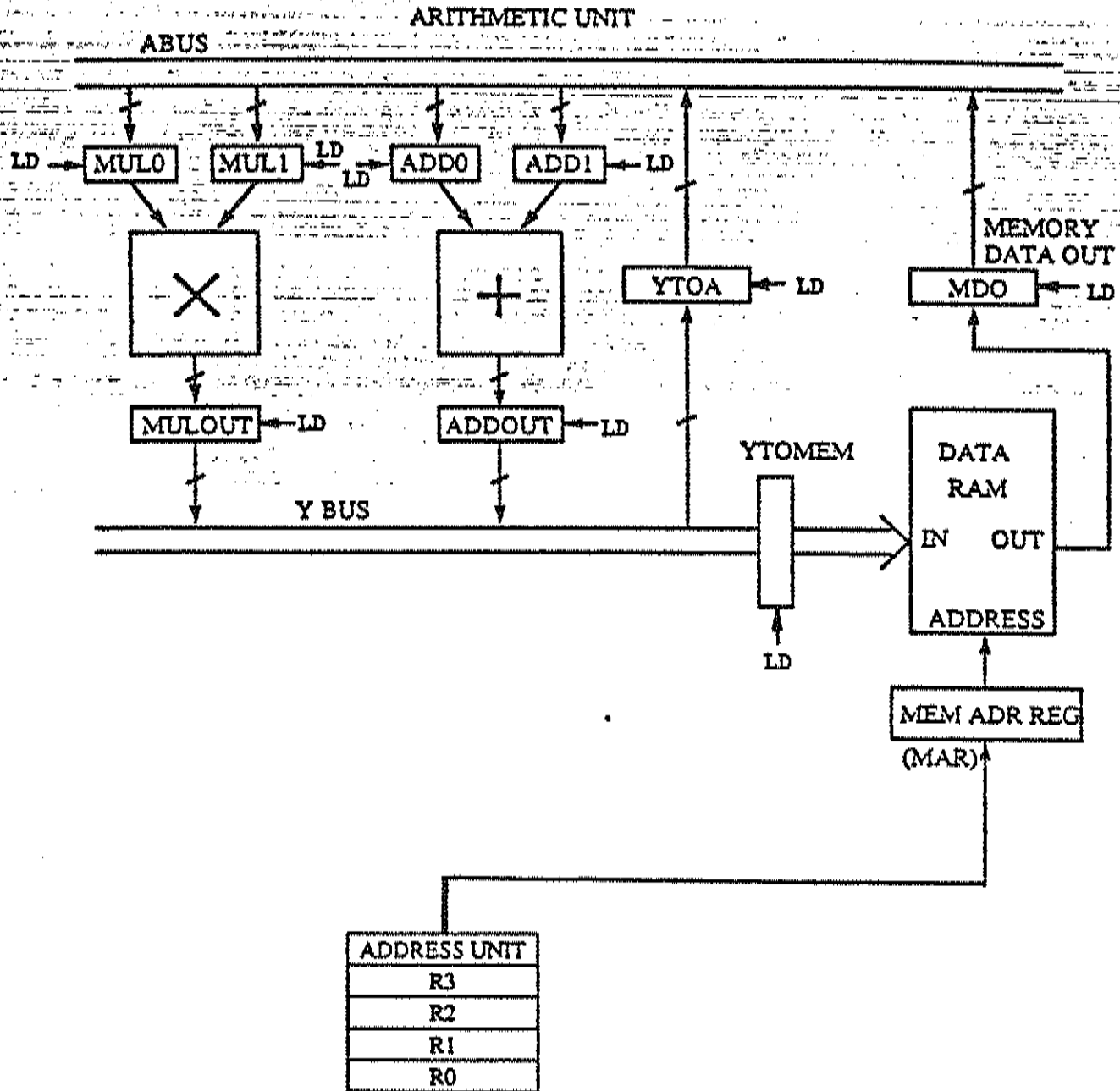
NOP

NOP

NOP

MULOUT → YTOMEM; write into (R0).

Assume Horizontal Microcode -- Assume all data regs are initially 0.



ADDRESS UNIT
R3
R2
R1
R0

(A) Give RTL description for the instruction

ADD (R0),(R1),(R2) which performs $(R0) + (R1) \rightarrow (R2)$

The two operands are pointed to by R0 and R1 respectively, and the result is stored in the location pointed to be R2. Include comments as appropriate. Use the answer sheet on the next page.

(B) Give RTL description for the "dot" product of two vectors A and B

$$\vec{A} = (a_1, a_2, a_3, \dots, a_n), \quad \vec{B} = (b_1, b_2, b_3, \dots, b_n), \quad \vec{A} \cdot \vec{B} = a_1 b_1 + a_2 b_2 + a_3 b_3 + \dots + a_n b_n$$

R0 points to vector A. R2 is location of result,

R1 points to vector B. R3 is the length of the vector.

Maximize throughput by keeping all adders and multipliers busy.

A)

Program Step	Operation(s)	Comments	
0	R0 → MAR		
1	R1 → MAR, ADD0 → ADD0	load (R0) into adder, R1 and	
2	R2 → MAR, ADD0 → ADD1		
3	NOP		
4	↓		
5			
6			
7			
8			
9			
10			wait for mult to
11			complete
12		↓	
13		ADDOUT → YTO MEM	store result into (R2).
14	all DONE	Done	
15			
16			
17			
18			
19			
20			

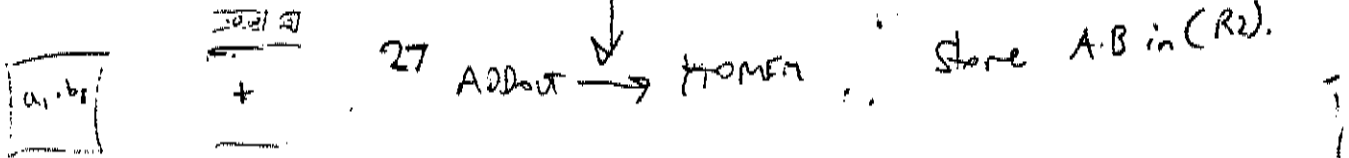
10/10

B)

Program Step	Operation(s)	Comments
0	R0 → MAR, R0++	print A, then increment
1	M0 → M0, R1 → MAR, R1++	print B, then increment
2	M0 → MUL1, ADDOUT → YTOA	start multiply, get previous
3	MULOUT → YTOA, YTOA → ADD0	add while mult's running
4	YTOA → ADD1	start addition
5	NOP	
6	NOP	
7	NOP	
8	NOP	
9	R3-1 → R3	decrement length counter
10	JUMP NOT ZERO, L000	loop back for next!
11	NOP	if end of vector finish f
12	NOP	wait for mult finish
13	MULOUT → YTOA	
14	YTOA → ADD0, ...	
15	ADDOUT → ADD1, R2 → MAR	all in last term
16	NOP *10	
17		wait for add to finish
18		
19		
20		

MULT. PLY. REMAINING 10 cycles

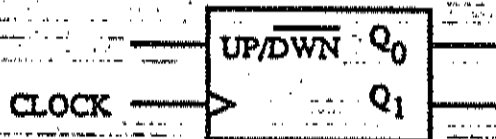
Can influence last add (R2=0).



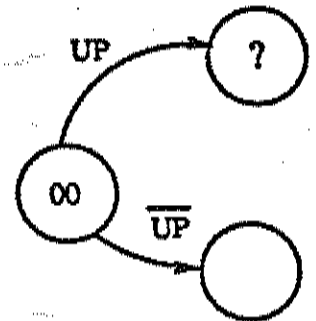
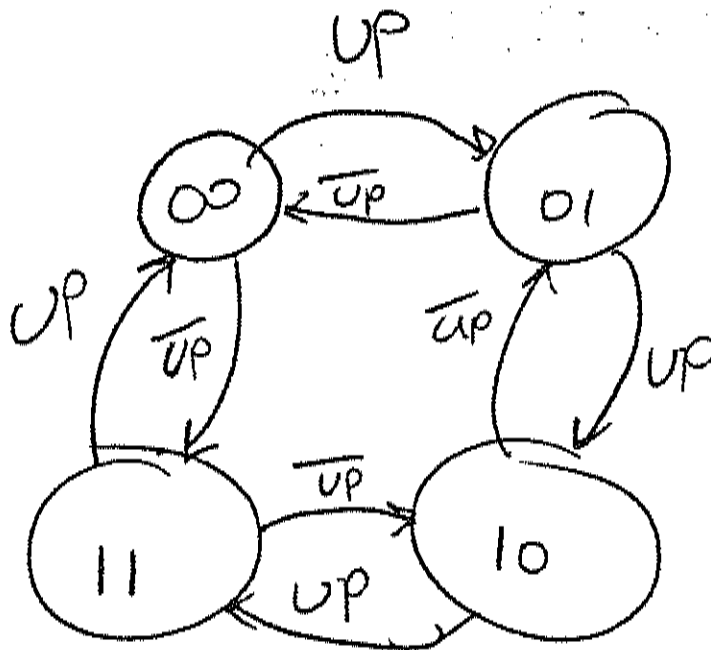
8

(15 min.)

In this problem you will design a synchronous 2 bit up/down binary counter using rising edge triggered JK Type Flip Flops. The output of the counter must be glitch free. The block diagram of the counter is:



(A) draw a state diagram for the counter, using notation shown to the right as an example.



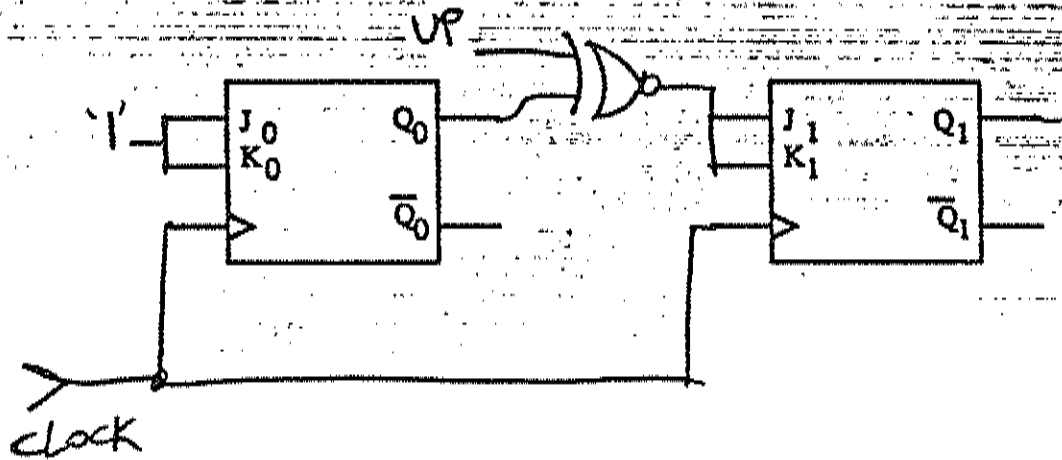
(B) Fill in the next state table for the up/down counter.

	PRESENT STATE		INPUT	CONTROLS				NEXT STATE	
	Q_1	Q_0	Up/Down	J_0	K_0	J_1	K_1	Q_1	Q_0
0	0	0	0	0	X	1	X	1	1
1	0	0	1	1	X	0	X	0	1
2	0	1	0	X	1	0	X	0	0
3	0	1	1	X	1	1	X	1	0
4	1	0	0	1	X	X	1	0	1
5	1	0	1	1	X	X	0	1	1
6	1	1	0	X	1	X	0	1	0
7	1	1	1	X	1	X	1	0	0

Note: Q_0 just toggles

Q_n	Q_{n+1}	J	K
0	0	0	0 X
0	1	1	0 X
1	0	0	1
1	1	X	0

8(Cont.) Complete the schematic of the counter, using minimized logic.



k-maps (for your convenience)

$$K_0 = \overline{Q_1} + Q_1$$

$$J_0 = \overline{Q_1} + Q_1$$

UP

	$Q_1 Q_0$	01	11	10
$\overline{Q_0}$		Φ	X	X
Q_0		1	X	X

UP

	$Q_1 Q_0$	01	11	10
$\overline{Q_0}$		X	1	X
Q_0		X	Φ	X

$$J_1 = \overline{Q_0} + Q_0$$

$$\overline{Q_0} \cdot \overline{UP} + Q_0 \cdot UP$$

UP

	$Q_1 Q_0$	01	11	10
$\overline{Q_0}$		1	0	X
Q_0		0	1	X

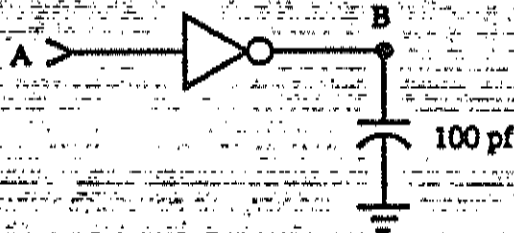
UP

	$Q_1 Q_0$	01	11	10
$\overline{Q_0}$		X	X	0
Q_0		X	X	1

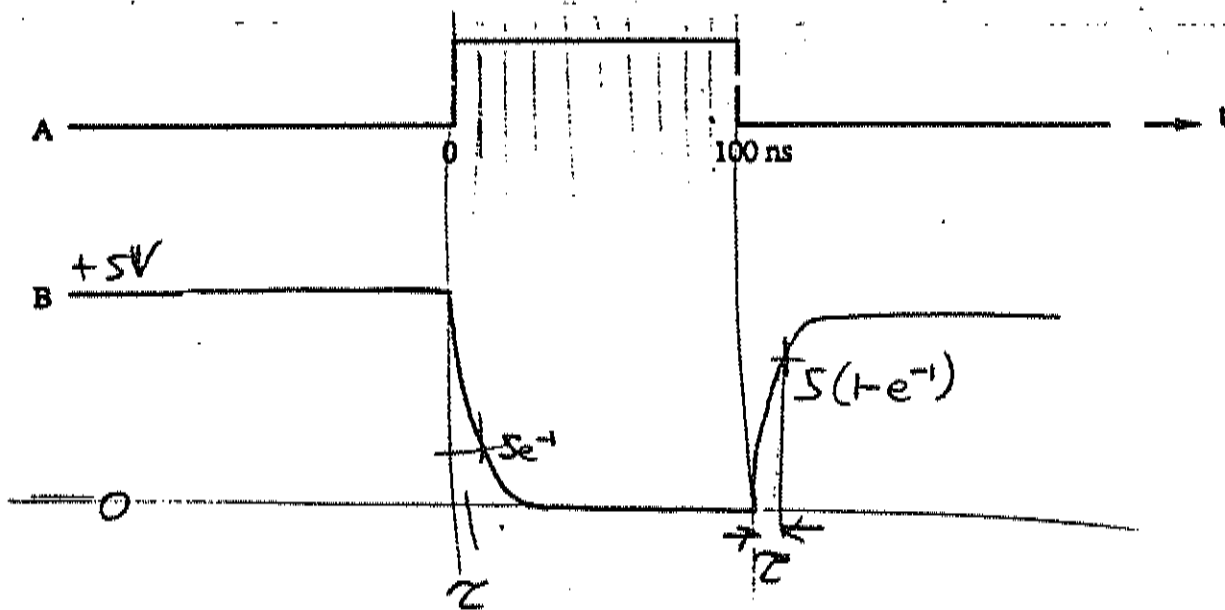
9

(5 min.)

Consider a logic device with 100Ω source impedance driving a 100 pF capacitive load. With no load, the driver switches between 0.0 and 5.0 volts DC.



Neglecting internal propagation delay through the inverter, complete the following sketch.



$$\tau = RC = (100\Omega)(100 \times 10^{-12}\text{ F})$$

$$RC = 10^{-8}\text{ sec} = 10\text{ ns}$$

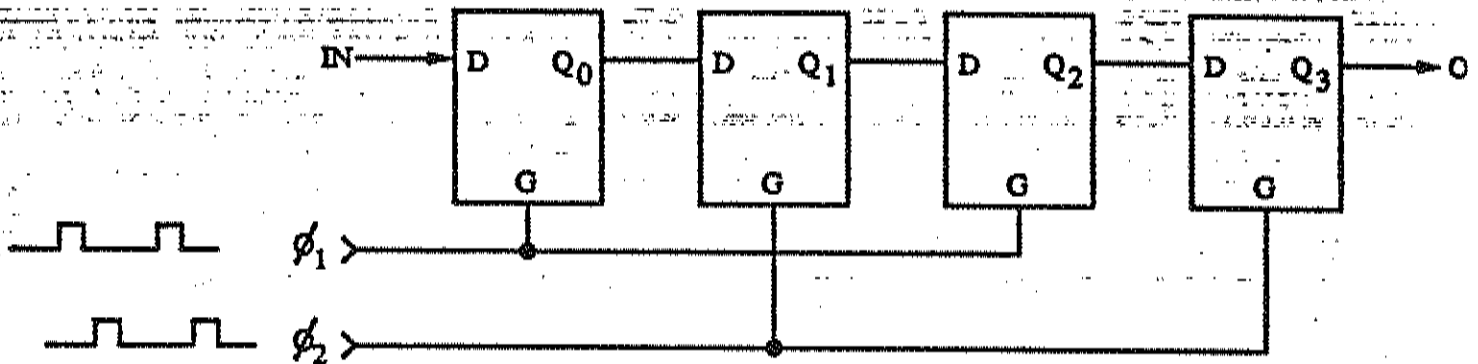
for 63%

$$V = 5(1 - e^{-1})$$

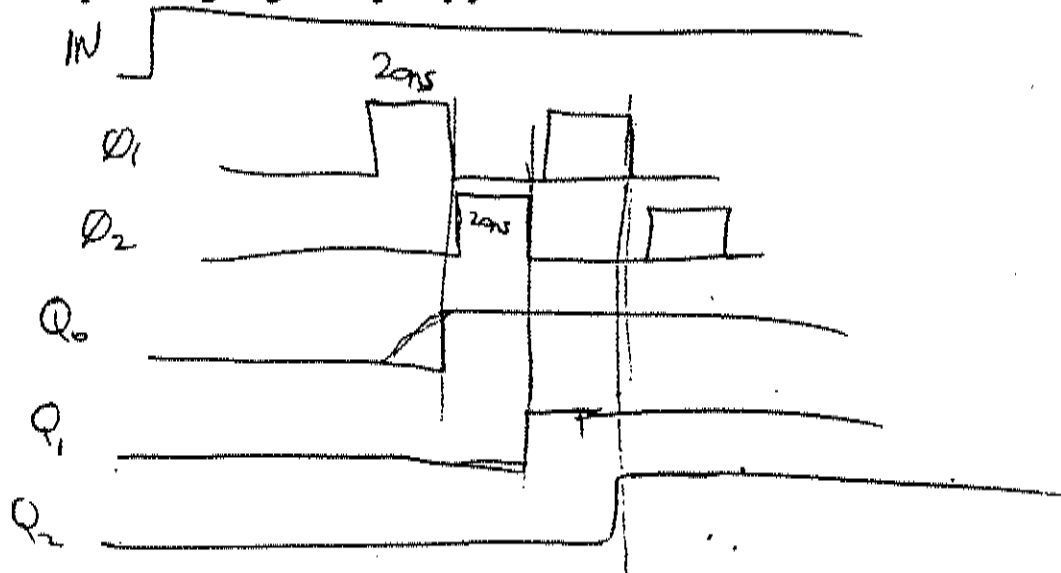
10

(15 min.)

Below is a 4 bit shift register using a two phase clock, and level-sensitive (not edge triggered) type D latches. The two clock phases are strictly non-overlapping ideal rectangular clocks. The latches have a 20 ns maximum delay from data-in to data-out. (Hold time from when the clock goes high is also 20 ns).

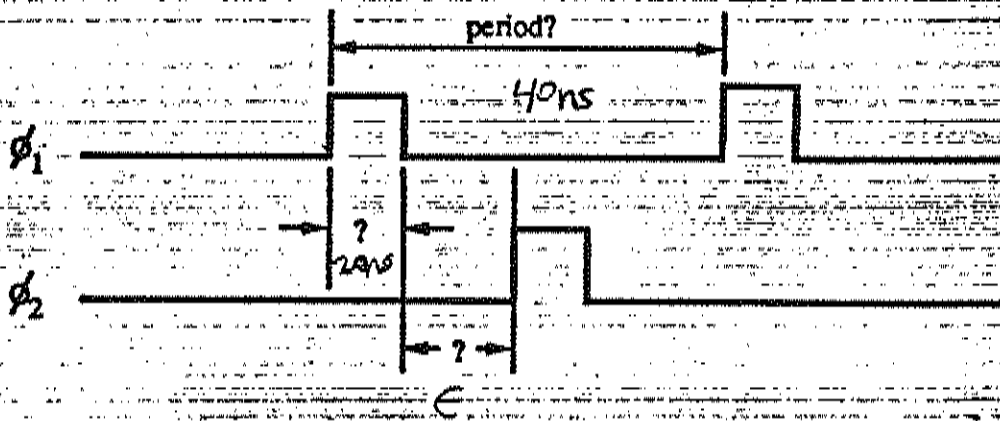


- (A) With no clock skew, what is the maximum frequency of operation? Give a simple timing diagram to justify your answer.

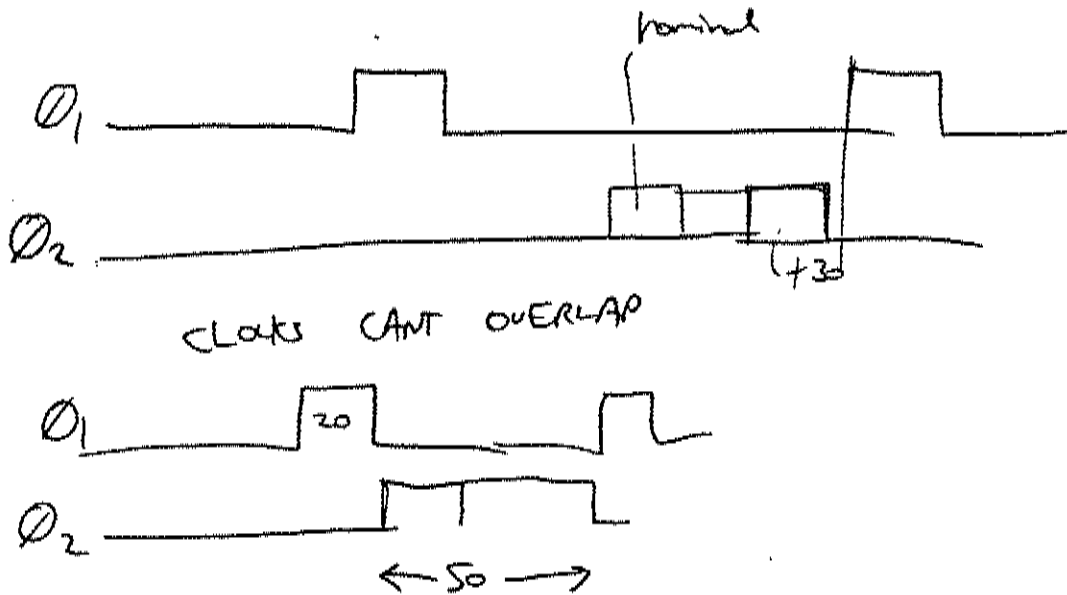


$$f = 1/40\text{ns} = 25\text{MHz}$$

(B) Specify clock parameters for part A.



(C) If ϕ_2 can be skewed (-0 ns, +30 ns) with respect to ϕ_1 , what is the maximum frequency of operation? Give a simple timing diagram to justify your answer.



$$f = \frac{1}{70\text{ns}} = 14\text{MHz}$$

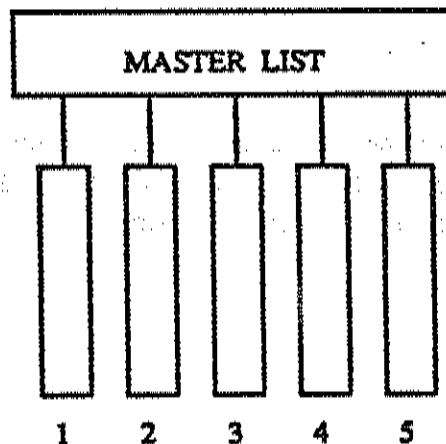
11

(30 min.)

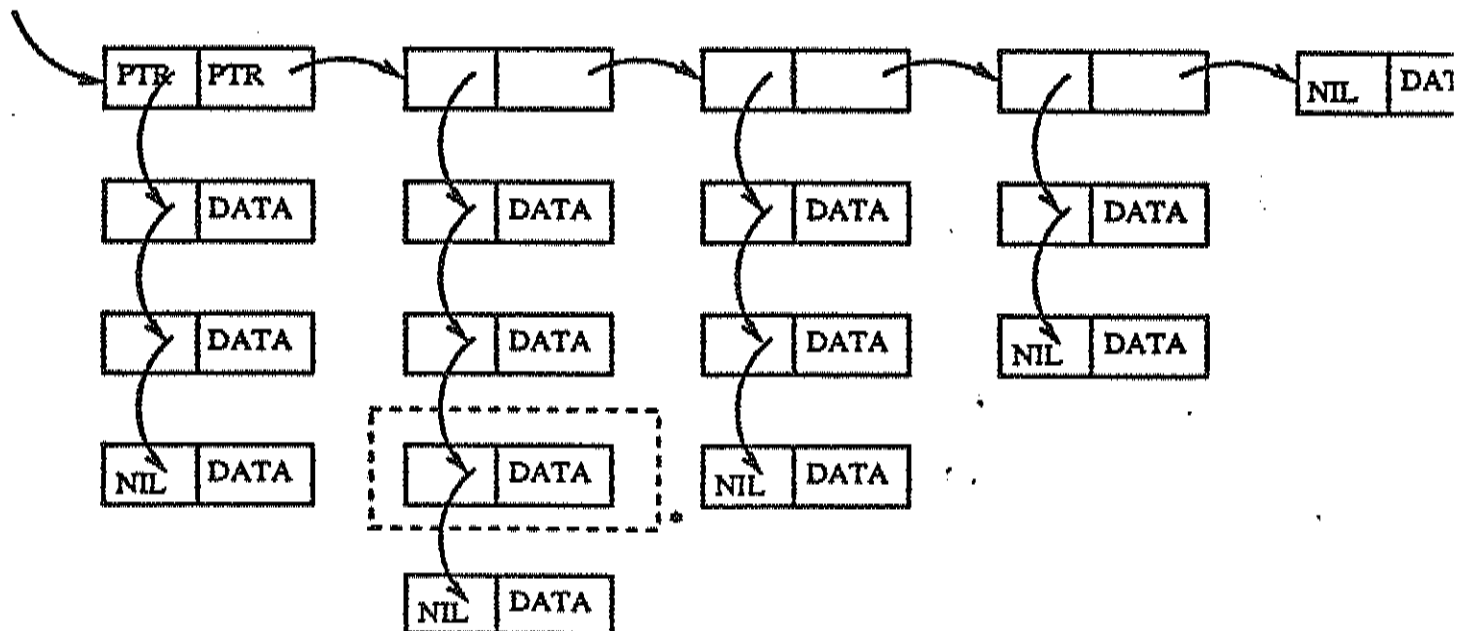
In this problem, you will be doing the high level design for a list processor. The design is built around a 64K x 16 memory which contains a "list of lists". The internal organization is shown below:

A cell is composed of two 16 bit words. For the master list, the first element in the cell points to a sublist, the second element points to the next cell in the master list. For the sublist, the first element in the cell is a pointer to the next cell, and the second element is a 16 bit word of data. A special pointer value of 'NIL' is used to indicate the last entry in a list.

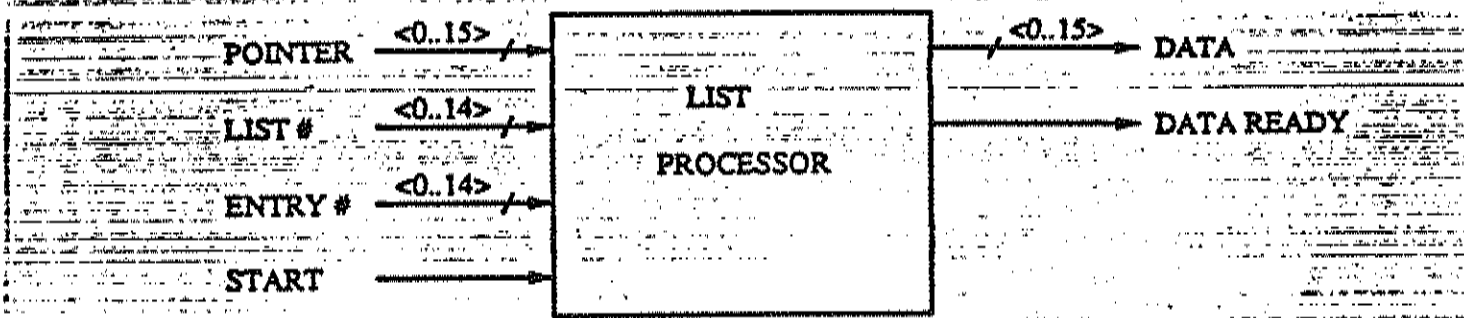
The third entry of the second list is marked with an asterisk.



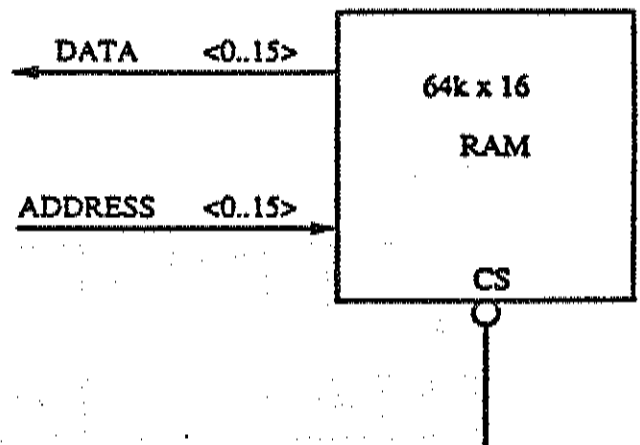
MASTER LIST



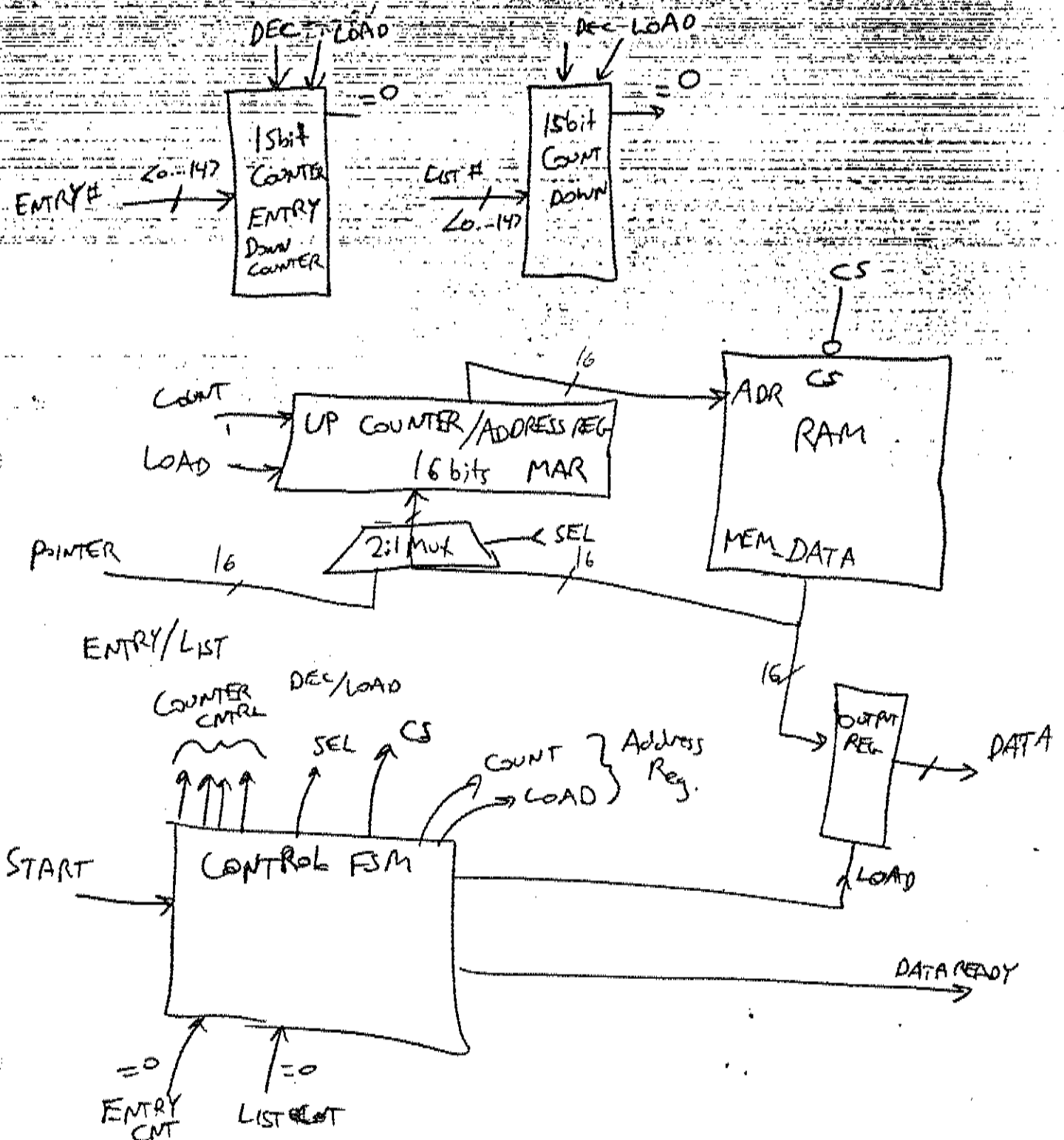
Operation of list processor:



Pointer is address of root of list. LIST and ENTRY specify which entry of which sublist to return. On START assertion, the List Processor begins to search down the master and sublist. Once the appropriate data is found it is sent out on DATA $\langle 0..15 \rangle$ and DATA-READY is asserted. Assume that a NIL will not be encountered before the desired entry is found.



(A) Give a block diagram implementation of such a device, using MSI components: registers, counters, flip flops, multiplexors, adders, comparators, etc., and a 64K x 16 RAM. Choose a straight-forward approach; no need to be fancy or show any details, e.g., you may just show one block for a "control FSM", and you don't need to show any gate level details.



(B) SPECIFY operation of the list processor using RTL.

1. Initialize/load values

POINTER \rightarrow MAR
 ENTRY \rightarrow ENTRY COUNT
 LIST # \rightarrow LIST COUNT

2. get appropriate list

LIST_LOOP LIST-1 \rightarrow LIST

JMP ZERO, GOT_LIST ; appropriate list found

MAR+1 \rightarrow MAR ; 2nd entry in cell points

next list

MEM_DATA \rightarrow MAR ; get pointer to next cell

JMP LIST_LOOP

3. now get appropriate entry in sublist, MAR points to first entry

GOT_LIST MEM_DATA \rightarrow MAR

GOT_LIST ENTRY-1 \rightarrow ENTRY

JMP ZERO, GOT_ENTRY

JUMP GOT_LIST

GOT_ENTRY MAR+1 \rightarrow MAR ; get 2nd word of cell

MEMDATA \rightarrow OUTPUT

STATUS 1 \rightarrow DATA READY

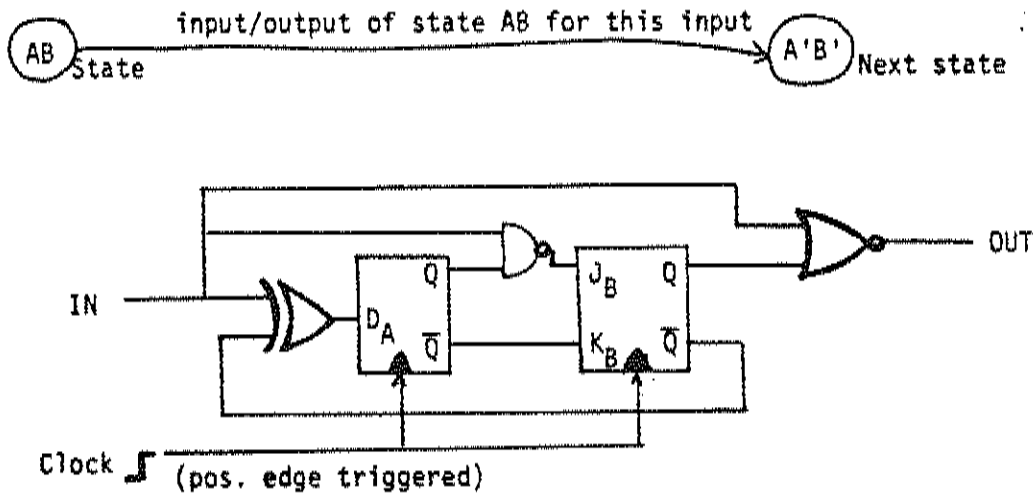
Done.

University of California
HARDWARE PRELIM EXAM - FALL 1985

General Instructions:

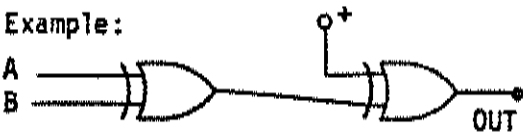
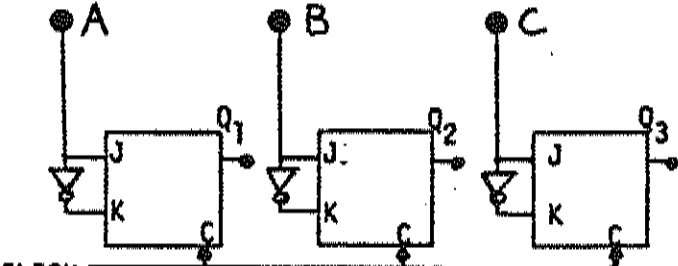
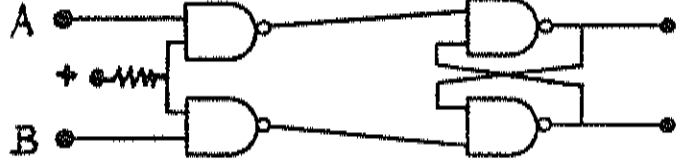
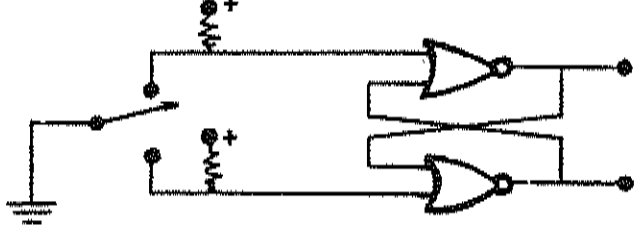
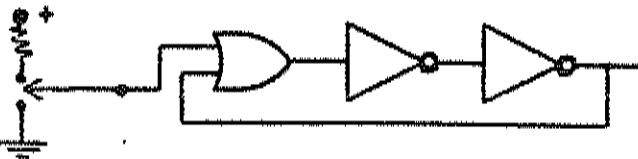
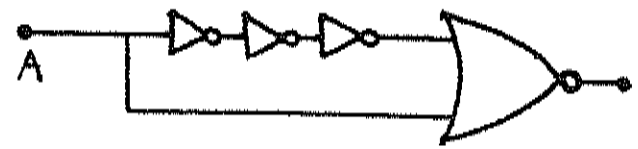
The exam lasts 3 hours = 180 minutes.
 Do all your work on these exam papers. (use back if necessary). Your Code
 Read the problem statements carefully.
 You should not need to ask questions.
 If something seems unclear, state your assumptions.
 The indicated points give you an idea how long a problem should take.
 (about one minute per point.)

1. Draw the complete state diagram (all states, all transitions) of the following Mealy machine containing two different positive edge triggered flip-flops.
 Label the transition arrows with the input/output values in the following style:



(15 points)

2. Match each circuit below with the one sentence that best describes it.

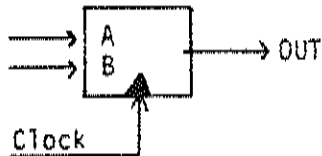
<p>Example:</p>  <p>Match 7</p>	
 <p>CLOCK positive edge-triggered flip-flops ○</p>	
 ○	
 ○	
 ○	
 ○	

1. RS latch.
2. Clocked RS flipflop.
3. Clocked D flipflop.
4. Clocked T flipflop.
5. Clocked JK flipflop.
6. Exclusive OR.
7. Exclusive - NOR.
8. Serial shift register.
9. Parallel (shift) register.
10. Up counter.
11. Down counter.
12. Ring or Moebius Counter.
13. Clock.
14. Clock with ON/OFF input.
15. Pulse shaper: outputs a single short pulse in response to the rising edge of an input.
16. Pulse shaper: outputs a single short pulse in response to the falling edge of an input.
17. Edge detector: outputs a single short pulse in response to a change in the input: (0 to 1) or (1 to 0).
18. Device to detect and remember the occurrence of a single positive input.
19. Device to detect and remember the occurrence of a single zero-going input.
20. Debounced switch.
21. None of the above.

(20 points)

3. Formal Problem Statement

Draw the complete state diagram for the following Moore machine:
A finite state machine accepts synchronously with the clock data on the single-bit wide inputs A, B. The machine has a single output that is at logical 1 when the data on the two inputs has been the same ($A=B$) for 3 or more consecutive clock cycles; otherwise it is at logical 0.



Draw the complete state diagram.

(15 points)

4. A network has 4 inputs R, S, T, U and 4 outputs V, W, Y, Z. RSTU represents a binary coded decimal digit, VW represents the quotient, and YZ the remainder when RSTU is divided by 3. First, realize the network using

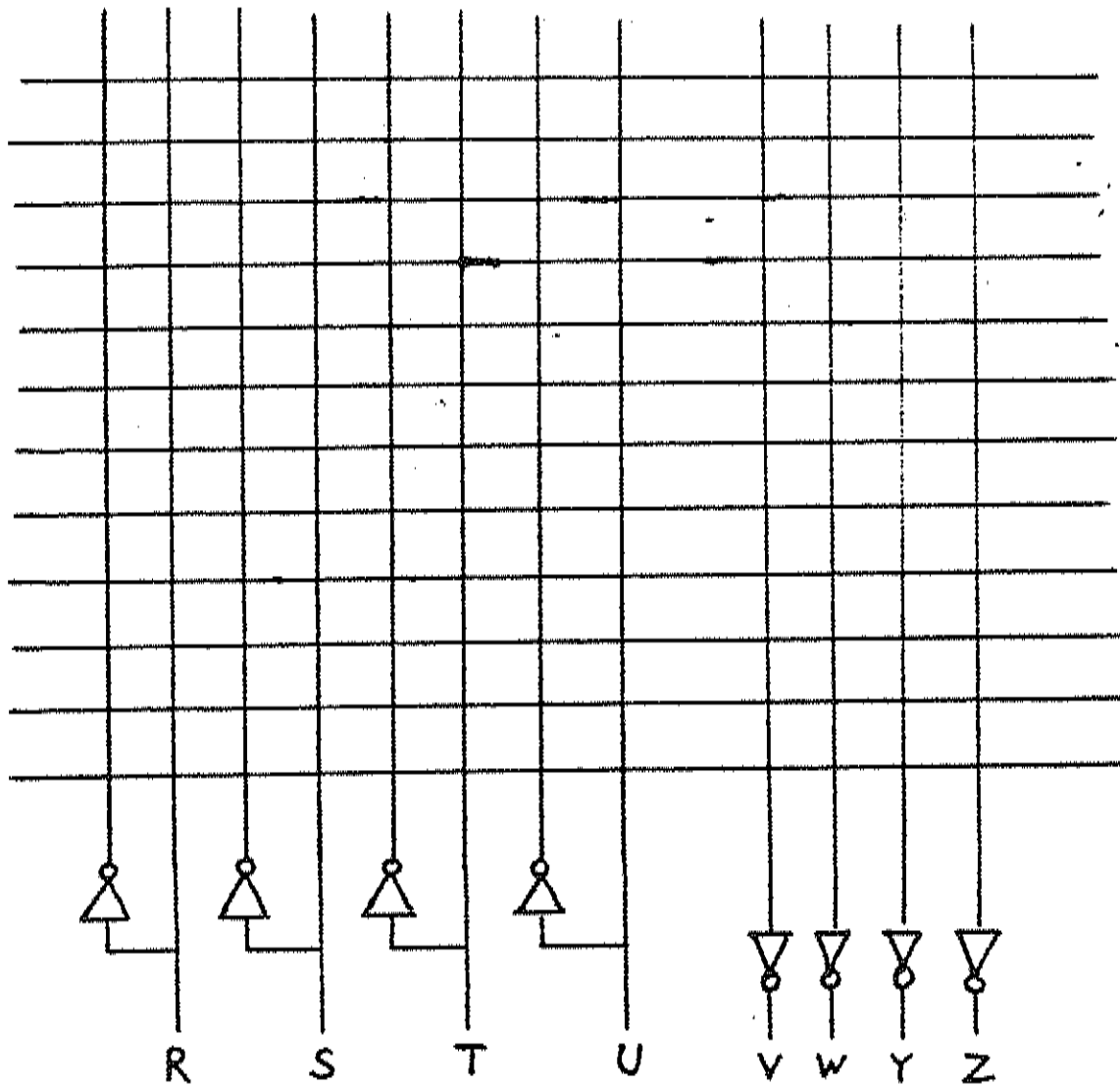
a) A 'minimum' 2-level NOR-gate network (not counting inverters). Verify your design by showing in your final circuit diagram the logical values on all lines when the inputs represent digit "3".

(25 points)

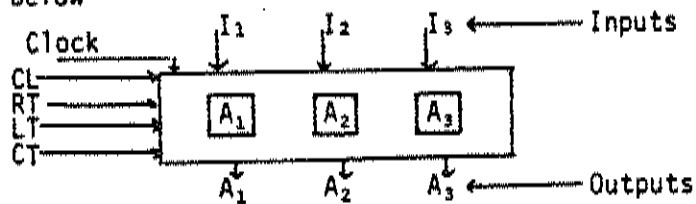
4. (continued)

Now implement the same network as a NOR-NOR-PLA (not just a ROM) using the grid below. Verify your design by showing on each signal line the logical value when the input is the digit "8".

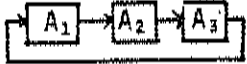
(15 points)



5. A_1 , A_2 and A_3 are 3 clocked D flipflops which act as a register. The register performs a specific operation depending upon the control signal present at the beginning of a clock pulse. The schematic of the whole register as well as the control signals and the corresponding function are given below



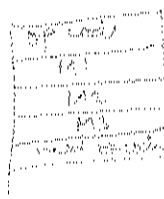
Assume exactly one control signal is present at any time.

<u>Control Signal</u>	<u>Operation</u>
CL	A_1, A_2, A_3 are reset to "0" (cleared)
RT	Contents of A_1, A_2, A_3 are shifted right one place end-around 
LT	Contents of A_1, A_2, A_3 are shifted left one place end-around
CT	The binary number $A_1 A_2 A_3$ is incremented by 1 (count up). mod-8. e.g., $(111) + (001) = (000)$.

The absence of any control signal implies a LOAD.

- (a) Derive the combined flip-flop excitation (input) equation for each flip-flop to perform the given operations. Simplify them as much as you can.
- (b) How can you prevent more than one control signal occurring simultaneously? Show two non-trivially different solutions with a corresponding circuit.

(30 points)





6. The most general form of an arithmetic instruction is

op 11,12,13,14

with 11 specifying the location in which to store the result of the binary operation (op) on the contents of the operand locations 12 and 13, and 14 specifying the location of the next instruction. The action of the instruction is of this form:

11 ← 12 op 13; goto 14

Computers are generally classified as being 0-, 1-, 2-, or 3-address machines. Your task is to assign values to 11, 12, 13, and 14 that transforms this general form into the four styles of machines.

# addresses	11	12	13	14
zero	[SP+1]	[SP]	[SP+1]	PC+1
one	ACC	ACC	[M]	PC+1
two	[M2]	ACC	[M2]	PC+1
three	[M1]	[M2]	[M3]	PC+1

Fill in the table either by using the following registers (or registers plus or minus a constant)

- PC (program counter)- contains the address of the current instruction (PC + 1 is the address of the next sequential instruction)
- SP (stack pointer)- points to the top of the stack. (This stack grows down, with SP + 1 pointing deeper in the stack and SP - 1 pointing outside the stack.)
- ACC (accumulator)

or by using the operand addresses of your choice, such as A1, A2, A3,

Remember, your assignment of registers or addresses or both to this matrix should transform the general form of the instruction into the zero, one, two, or three address form.

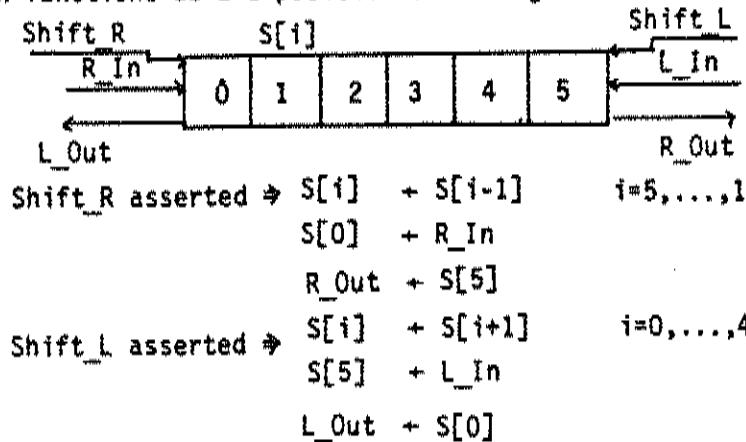
(15 points)

Handwritten note: needs to be 11, 12, 13, 14



Handwritten note: done

7. A hardware stack is to be used in an 18-bit small computer. The hardware stack is to be implemented using an LSI building block which functions as a 6-position shift register:



- (A) How are stack overflow and stack underflow error conditions sensed by the stack control logic? (Hint: you may only use shift registers and comparison logic.)
- (B) What modifications need to be made to the shift register building block so that an executing program could "peer into the stack" without disturbing the top of the stack (TOS)? You may respecify only the outputs of the building block.
- (C) Design the control logic to the gate level to implement the ability to "peer into the stack."

(25 points)

8. Consider a processor that dedicates a separate stack in main storage to service subroutines exclusively; another stack in main storage is dedicated exclusively to servicing interrupts.
- (A) Describe the control logic that would allow the interrupt stack to build on the subroutine stack if the interrupt stack overflowed. How would the processor know it was administering an interrupt in the subroutine stack? How would it know when to leave the subroutine stack because it had exhausted the interrupts that had been nested there and return to the interrupt stack to unwind interrupts nested there?
- (B) Is (A) a realistic approach to handling a heavy interrupt load? With the cost of commercial RAM sufficiently low, wouldn't it be simpler to merely extend the area of main storage allocated to the interrupt stack? Is there a reasonable maximum size to a hardware stack allocated exclusively to interrupts (i.e., how many words of main storage should be dedicated to the interrupt stack so that the probability of a stack overflow error occurring is small)?

We expect a brief essay-type answer for each part. It is not necessary to give a gate level design.

(20 points)

Hardware Core Exam: Spring 1986

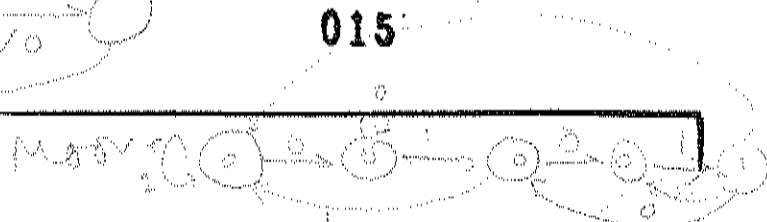
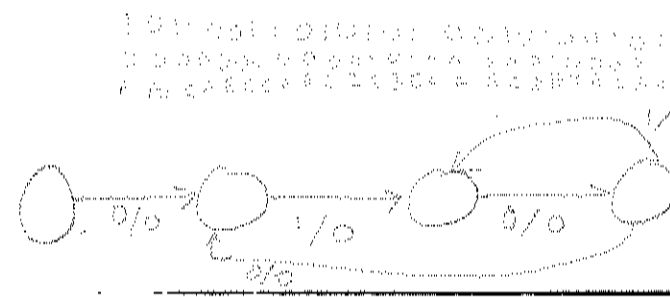
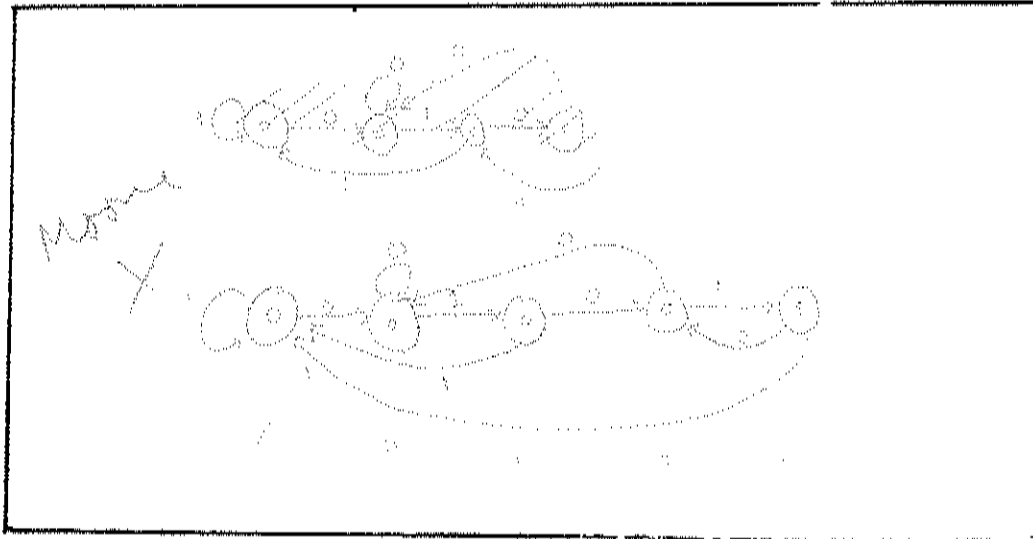
PROBLEM 1

(pages 1 - 3)

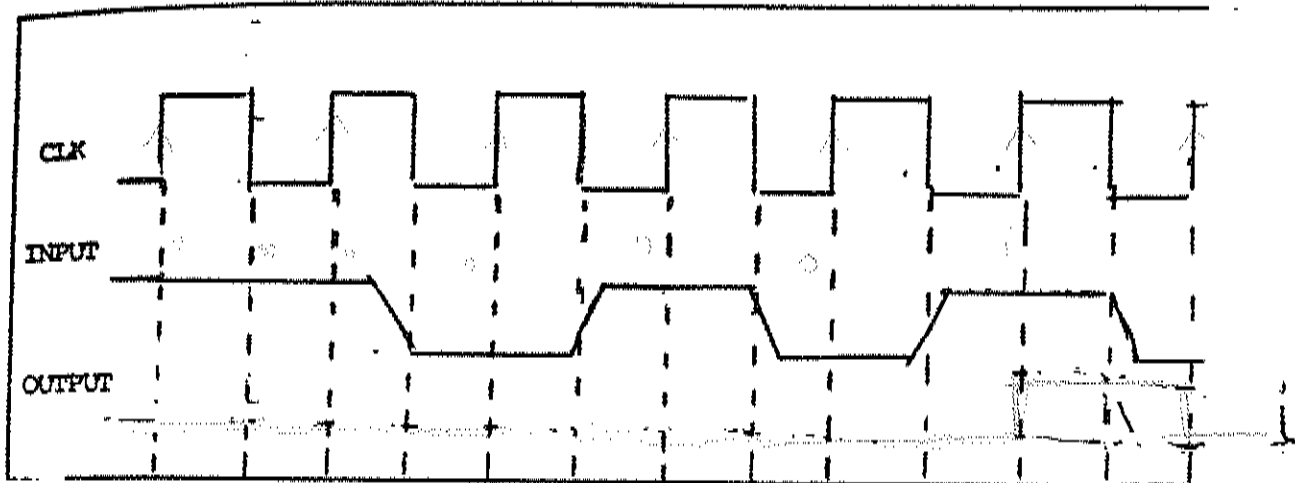
Construct a Mealy Model synchronous sequential machine that implements a sequence detector, as specified below:

1. The machine has one external input line, one external output line, and a clock input.
2. The machine is to detect all occurrences of the input sequence 0101 (i.e., the output signal is 1 only if the last four input signals were 0101)
3. The machine is to have a minimum no. of internal states.
4. Assume that the detector starts (arbitrarily) in state A.

Draw a state diagram for the sequence detector described above:

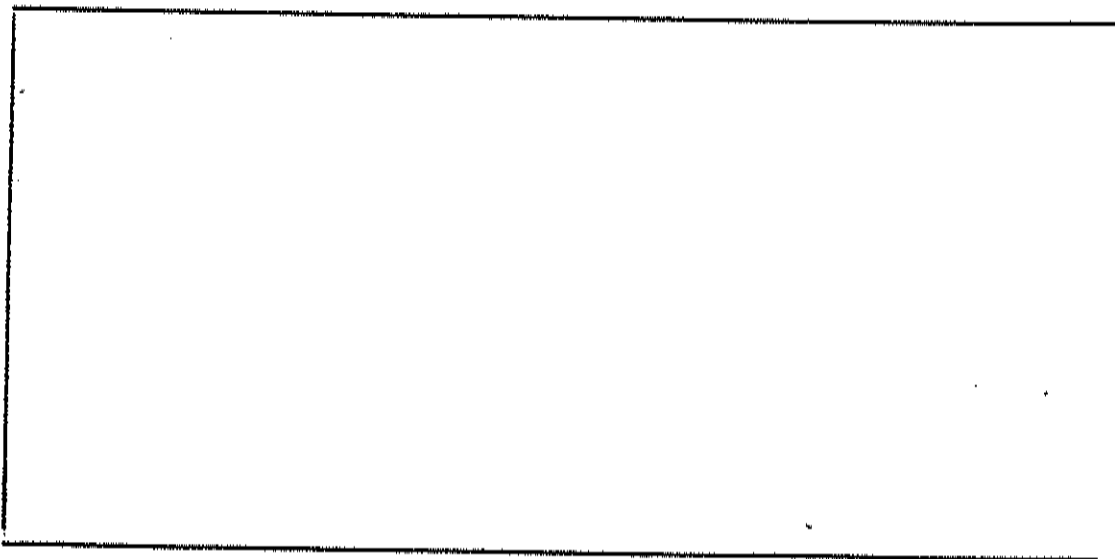


Complete the timing diagram below by drawing the output waveform (transitions occur on the rising edge of the clock probe).



Construct an encoded transition table, show FF excitation tables, and draw the logic circuits required to implement the sequence detector, using J-K flip flops.

(You may continue your solution on Page 3)



PROBLEM 2
(pages 4 - 6)

An M-N flip flop works as follows:

- If MN = 00, the next state of the flip flop is 0
 If MN = 01, the next state of the flip flop is the same as the present state
 If MN = 10, the next state of the flip flop is complement of the present state
 If MN = 11, the next state of the flip flop is 1.

- a) Complete the following table (use don't cares when possible)

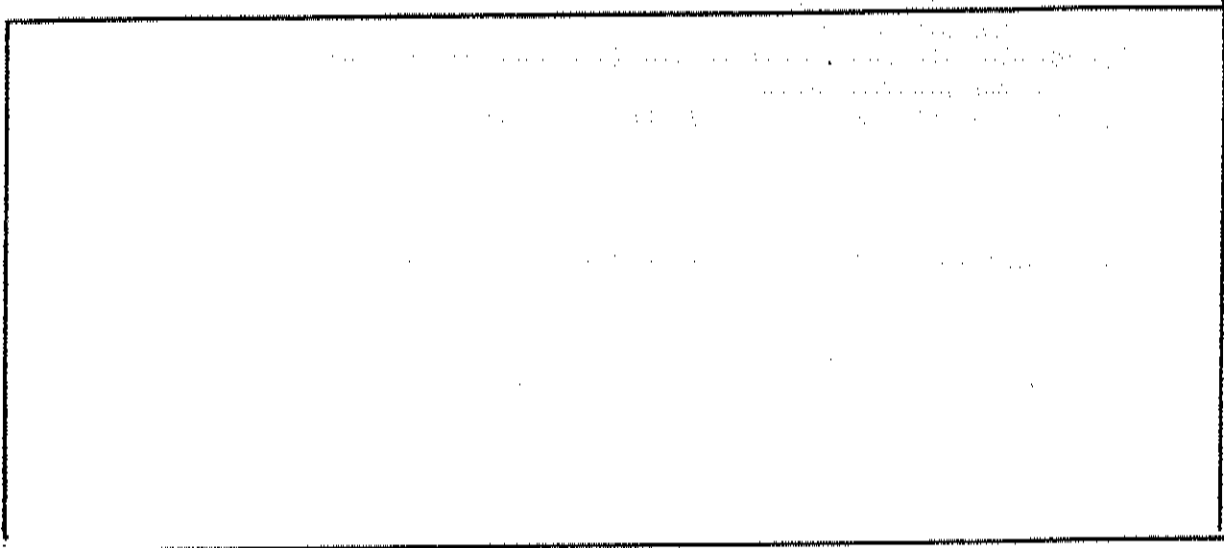
Present State Q	Next State Q ⁺	M N
0	0	
0	1	
1	0	
1	1	

- b) Using the above table and Karnaugh maps, derive and minimize the flip flop input (excitation) equations for a counter composed of 3 M-N flipflops (Q_1, Q_2, Q_3) which counts in the following sequence:
 $Q_1 Q_2 Q_3 = 000, 001, 011, 111, 101, 100, 000, \dots$

(Please show all work on Page 5)

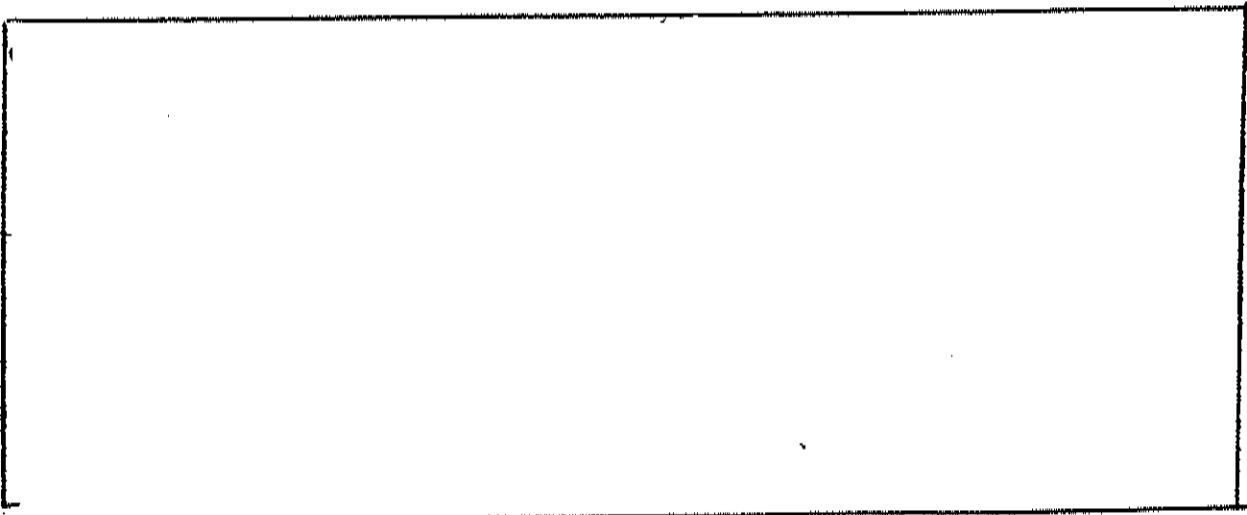
Problem 2

- c) How would you make this counter lock-out free?
Give an approach.



- d) Test the correctness of your solution to part b) for
the following transitions:

001 to 011 and 000 to 001



PROBLEM 3
PAGES 7 - 12)

This question requires you to complete a microprogram for the computer described on the next 5 pages. The answers must be copied into the place provided on this page below. The description of the computer includes

- A incomplete microprogram,
- A description of the microinstruction format,
- A finite state machine,
- A block diagram, and
- An instruction set description in a hardware description language.

The attached description is meant to be complete, but you may need to read other sections of the microprogram to determine how the machine really works.

There are 15 blanks that must be filled in: 3 state numbers, 8 microinstructions, and 4 comments that must be written completely (microinstruction 8) or just finished (microinstructions 18, 19, and 20). **YOU MUST WRITE YOUR ANSWERS ON THIS PAGE.** The blanks to be filled in are shown as _____, and the right hand column lists the number of blanks per line.

State	Micro Addr.	Microinstruction	Comment	(No. Blanks)
1	3	<u>just ready</u>	Loop if not ready	(1)
...	...	_____	_____	(3)
...	8	_____	_____	(2)
...	9	_____	_____	(1)
10	14	_____	->OP=010:BRN	(2)
...	...	_____	_____	(2)
2	18	_____	DECODE:OP=_____	(2)
2	19	_____	DECODE:OP=_____	(2)
...	20	RT,MBR,AC	->OP=_____:STOR	(2)
...	...	_____	_____	(1)
13	25	_____	->OP=101:LOAD	(1)
...	...	_____	_____	(1)
6	32	_____	AC <- AC & M	(1)
...	...	_____	_____	(1)

TOTAL CORRECT: ___/15

STATE

MICRO ADDRESS

MICRO INSTRUCTION

COMMENT

0
0
1
2
3
4
5
6
7
8
9
9
2
2
10
10
11
11
2
2
10
4
4
4
2
13
13
5
2
14
14
6
6
15
15
7
7

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37

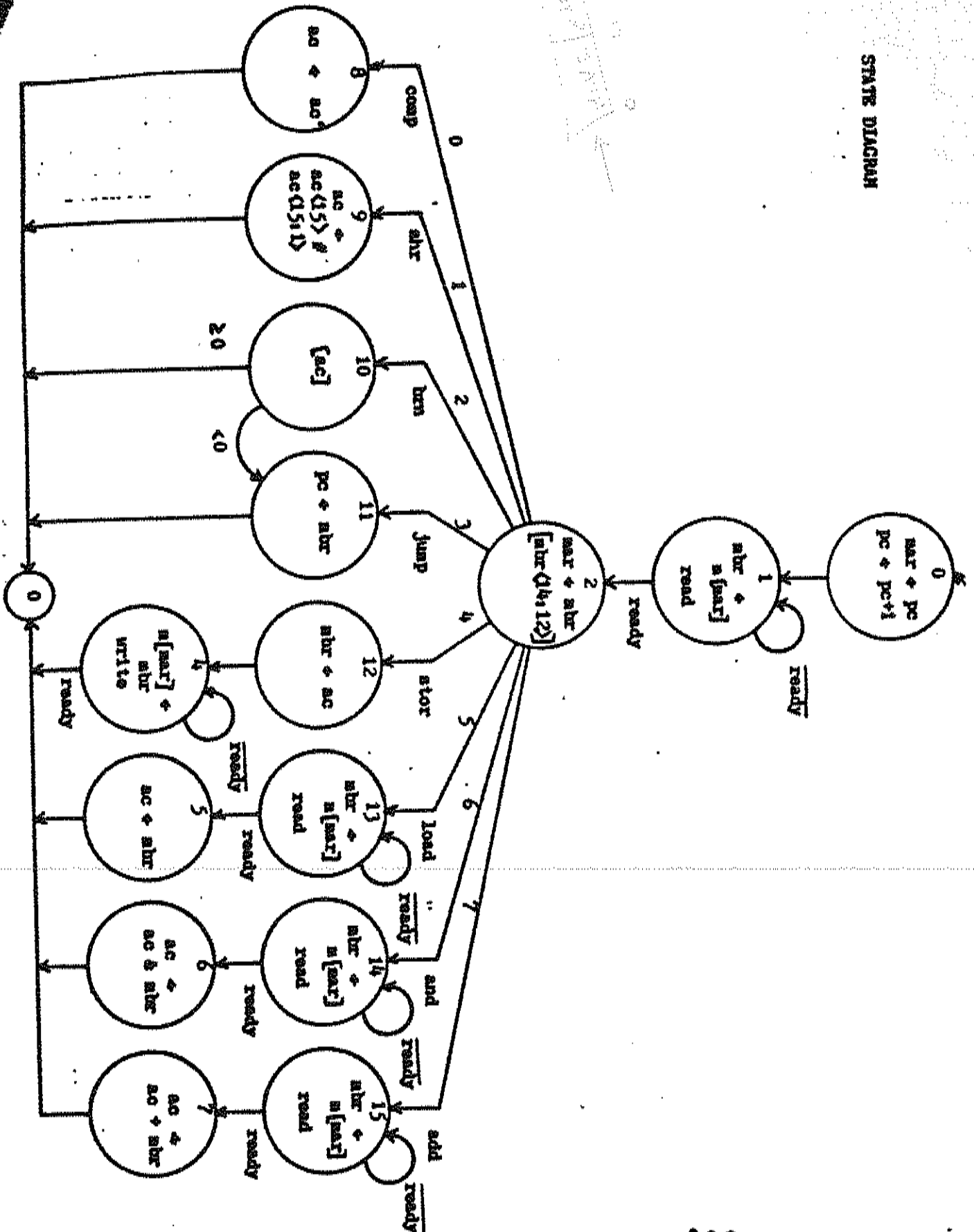
RT, MAR, PC
RT, PC+1
RT, MBR, M
TEST, READY, 0
RT, MAR, MBR
TEST, MBR<14>, 7
TEST, MBR<13>, 7
TEST, MBR<12>, 3
OP, COMP, AC
OP, ASHR
JMPZERO
JMP, 6
TEST, MBR<12>, 3
JMPZERO
RT, PC, MBR
JMPZERO
RT, MBR, AC
RT, M, MBR
TEST, READY, 0
JMPZERO
JMP, 5
TEST, READY, 0
OP, PASS, MBR
JMPZERO
TEST, MBR<12>, 5
RT, MBR, M
TEST, READY, 0
JMPZERO
RT, MBR, M
TEST, READY, 0
OP, ADD, MBR
JMPZERO

MAR ← PC
PC ← PC+1
MBR ← M[MAR]
LOOP IF NOT READY
MAR ← MBR
DECODE OPCODE
DECODE: OP=0XX
DECODE: OP=00X
→OP=001: ASHR
DECODE: OP=01X
→OP=010: BRN
→OP 011: JMP
DECODE: OP=10X
DECODE: OP=11X
→OP=100: STOR
WRITE AC
LOOP UNTIL READY
→OP=101: LOAD
LOOP UNTIL READY
AC ← M
DECODE: OP=11X
→OP=110: AND
LOOP UNTIL READY
AC ← AC & M
→OP=111: ADD
LOOP UNTIL READY
AC ← AC + M

MICRO-INSTRUCTION FORMAT FOR
FOR VERY VERTICAL MICROPROGRAMMING

type	format	microoperations								
RT	<table border="1"> <tr> <td>1</td> <td>1</td> <td>3</td> <td>3</td> </tr> <tr> <td>1</td> <td>0</td> <td>DES</td> <td>SOR</td> </tr> </table> <p> 0: PC 0: PC 1: MAR 1: MAR 2: MBR 2: MBR 3: M 3: M 4: AC 4: AC 6: PC+1 7: PCO </p>	1	1	3	3	1	0	DES	SOR	<p>$\mu PC + \mu PC + 1$</p> <p>DES + SOR</p> <p>PC + PC+1 PC + 0</p>
1	1	3	3							
1	0	DES	SOR							
OP	<table border="1"> <tr> <td>1</td> <td>1</td> <td>3</td> <td>3</td> </tr> <tr> <td>1</td> <td>1</td> <td>OPR</td> <td>REG</td> </tr> </table> <p> 0: PASS 0: PC 1: AND 1: MAR 2: ADD 2: MBR 3: M 4: AC 6: COMP 7: ASHR </p>	1	1	3	3	1	1	OPR	REG	<p>$\mu PC + \mu PC + 1$</p> <p>AC + AC (OPR) REG</p> <p>AC + \overline{AC} AC + AC/2</p>
1	1	3	3							
1	1	OPR	REG							
JMP TEST	<table border="1"> <tr> <td>1</td> <td>3</td> <td>4</td> </tr> <tr> <td>0</td> <td>J/T</td> <td>JADRS</td> </tr> </table> <p> 0: JMPZERO 1: JMP, JADRS 2: TEST, MBR<15> 3: TEST, MBR<14> 4: TEST, MBR<13> 5: TEST, MBR<12> 6: TEST, READY 7: TEST, (AC<0) </p>	1	3	4	0	J/T	JADRS	<p>JADRS = 2's complement number (+7... -8)</p> <p>$\mu PC + 0$ $\mu PC + \mu PC + JADRS$</p> <p>if TEST true then $\mu PC \rightarrow \mu PC + JADRS$ else $\mu PC \rightarrow \mu PC + 1$</p>		
1	3	4								
0	J/T	JADRS								

STATE DIAGRAM



©DIDL: "HIGH LEVEL DESCRIPTION OF SM-2 COMPUTER"

INPUTS SANITY;
OUTPUTS;

REGISTERS MAR<11:0>, MBR<15:0>, PC<11:0>,
AC<15:0>, M[0:4095]<15:0>;

MACROS: COMP := 0; "OPERATION CODE ASSIGNMENT"
SHR := 1;
BRN := 2;
JUMP := 3;
STOR := 4;
LOAD := 5;
AND := 6;
ADD := 7;

X := MBR<11:0>; "EFFECTIVE ADDRESS"
INST := MBR<14:12>; "CURRENT OP CODE"

END MACROS.

FETCH: "INSTRUCTION FETCH PROCEDURE"

MAR ← PC, PC ← PC + 1;

NEXT

MBR ← M(MAR);

END FETCH.

EXECUTE: "INSTRUCTION EXECUTION PROCEDURE"

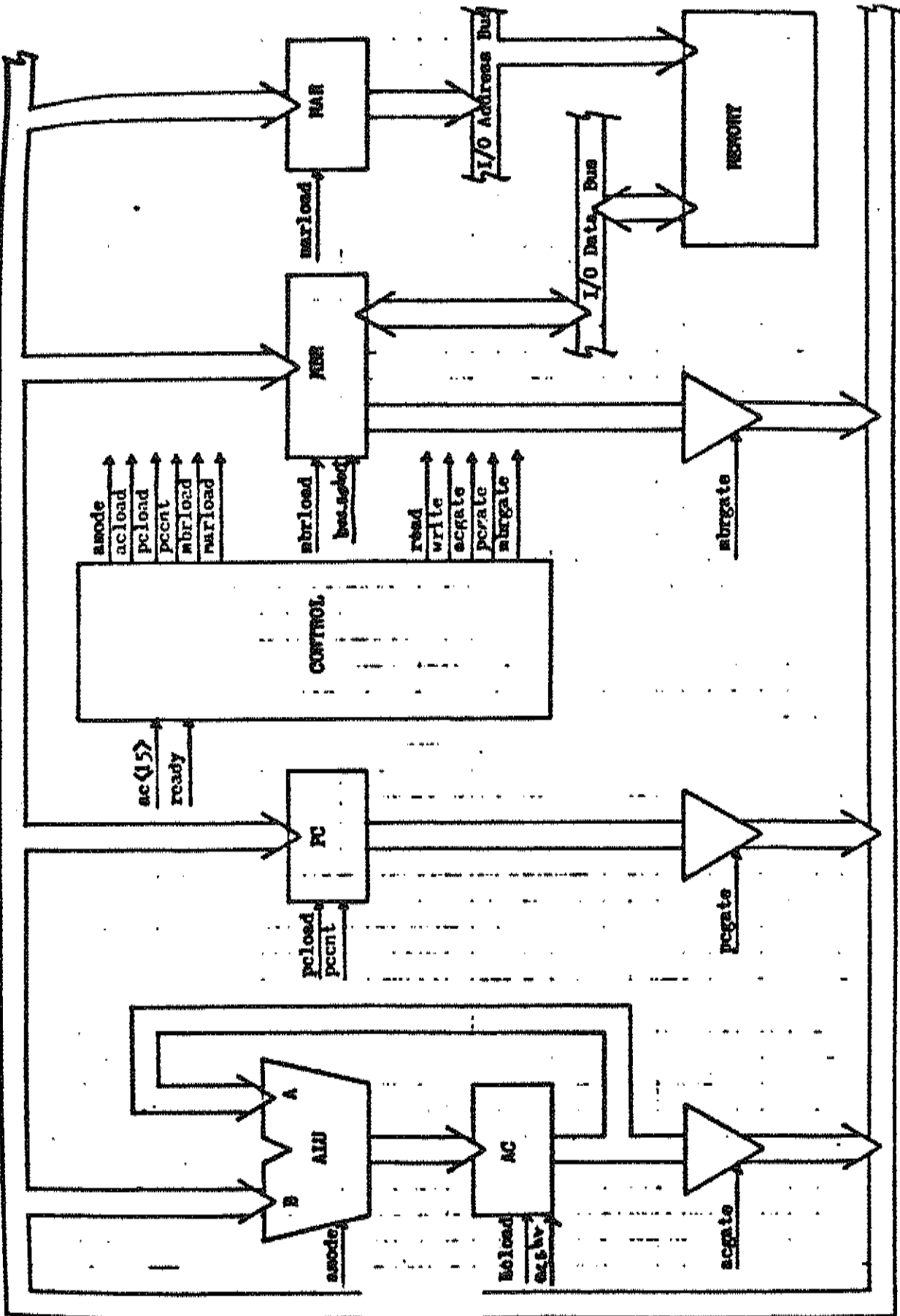
(INST = COMP) => AC ← AC;
(INST = SHR) => AC ← AC<15>#AC<15:1>;
(INST = BRN)&(AC<0) => PC ← X;
(INST = JUMP) => PC ← X;
(INST = STOR) => M[X] ← AC;
(INST = LOAD) => AC ← M[X];
(INST = AND) => AC ← AC & M[X];
(INST = ADD) => AC ← AC + M[X];

END EXECUTE.

SANITY => PC ← 0;

SANITY' => FETCH NEXT EXECUTE;

END.*



11-1-76

PROBLEM 4
(pages 13-14)

- a) (3 points). A processor-memory cache consists of a tag store and data store. Each entry in the tag store contains several pieces of information. Complete the table below by entering in X, Y, and Z pieces of information generally contained in a tag store entry and entering in A, B, C, D, E, F whether the corresponding piece of information is ALWAYS, SOMETIMES or NEVER present in the cache specified. Hint: one of the entries A,B,C, must be Never. one of the entries D,E,F must be Never.

Piece of Information	Present in Write-Through Cache	Present in Direct Mapped Cache
(X)	(A)	(D)
(Y)	(B)	(E)
(Z)	(C)	(F)

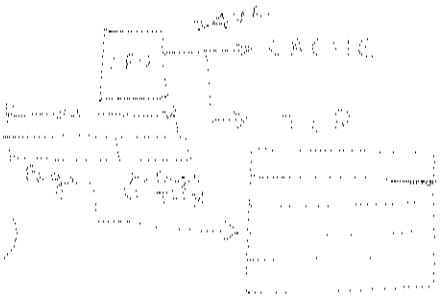
- b) (7 points). A cache has the following characteristics: 32 KB, 4-way ~~set~~ associative, physical cache, line size (i.e., block size) of 16 bytes. We wish to use this cache in a 16 MB paged virtual memory architecture that has 1 MB of physical memory. A TLB (Translation Look aside Buffer) is to be provided. (If you speak DEC terminology rather than IBM terminology, a TB is to be provided).

What is the minimum virtual memory page size necessary to allow accessing the TLB and the tag store of the processor-memory cache to be done concurrently?

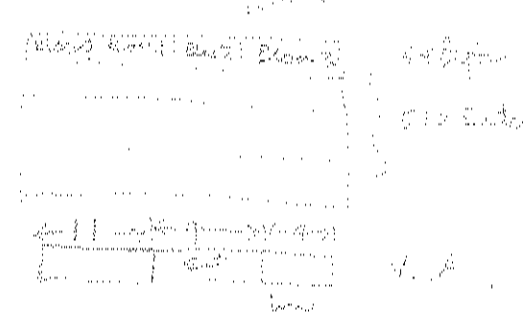
Answer: _____

Please show all your work on the next page.

32 KB cache, 4-way set assoc. line size 16 bytes
 16 MB virtual memory
 1 MB physical memory



32 KB
 16 sets of 4



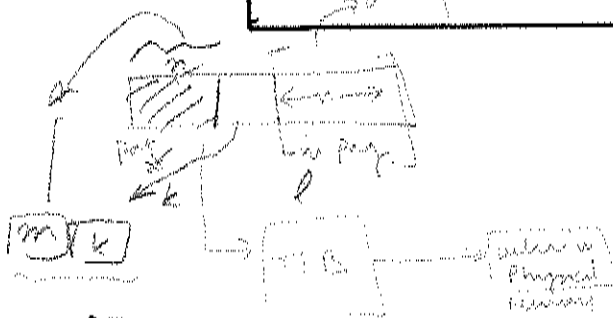
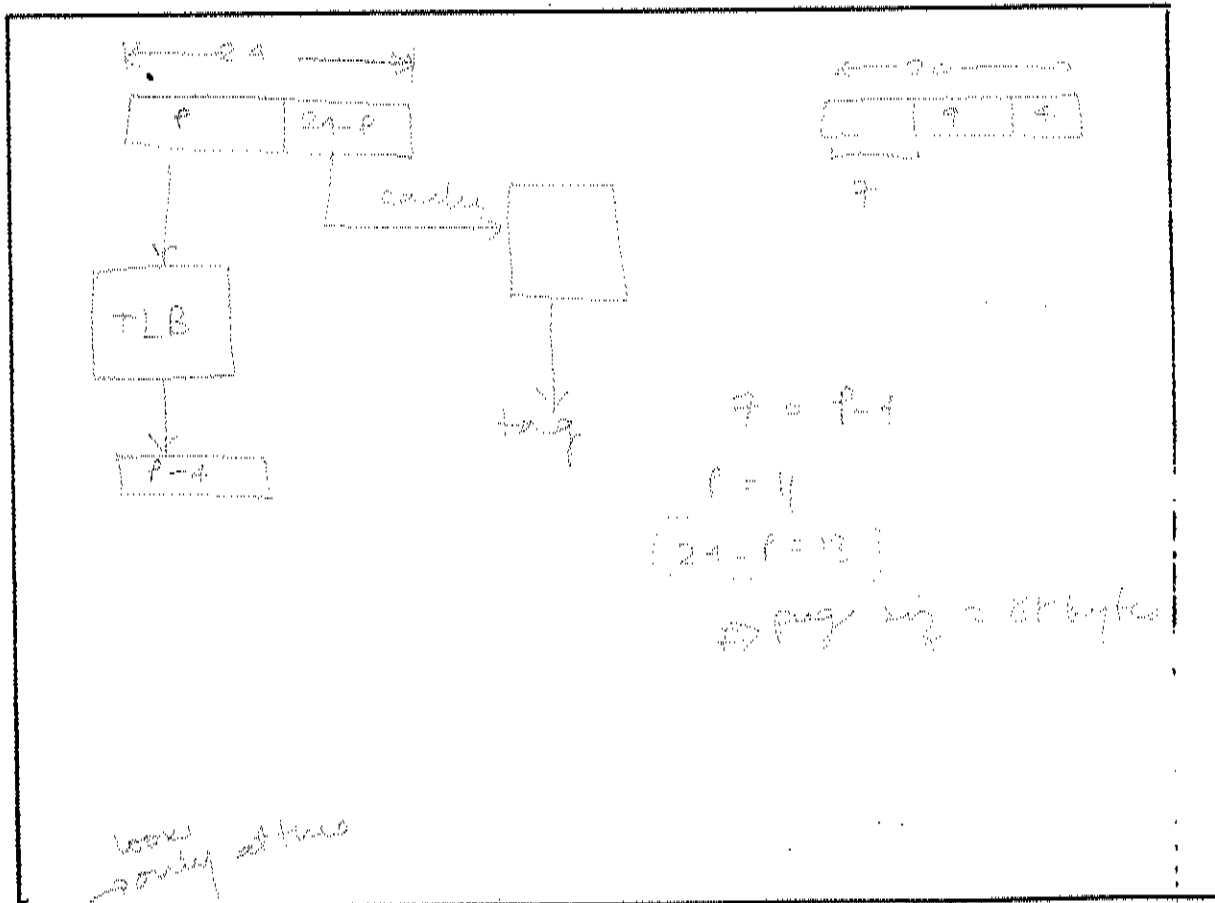
025

Page size = 2¹⁰ bytes
 2¹⁰ bytes
 Page size = 2¹⁰ bytes
 2¹⁰ bytes in TLB

PROBLEM 5
(page 15)

Consider a DMA device controller that can move an arbitrary length byte string to or from main memory in a single command.

Suppose addressing of main memory by the CPU is done through a conventional page table. (Thereby a conventional virtual memory is supported). Discuss the tradeoffs inherent in the decision of whether the DMA device will move byte strings to or from specific real addresses or specific virtual addresses



026

tag = physical page # + # of remaining bits

PROBLEM 6

(pages 16-25)

Problem #6 (20 points)

You are to design a computer. It is:

- a) a very simple, Von Neumann machine
- b) fully synchronous, with single cycle memory
- c) 12 bits per word
- d) to be constructed of ONLY NAND gates, of arbitrary fan-in and fan-out.

You are to describe your design 'top-down'.

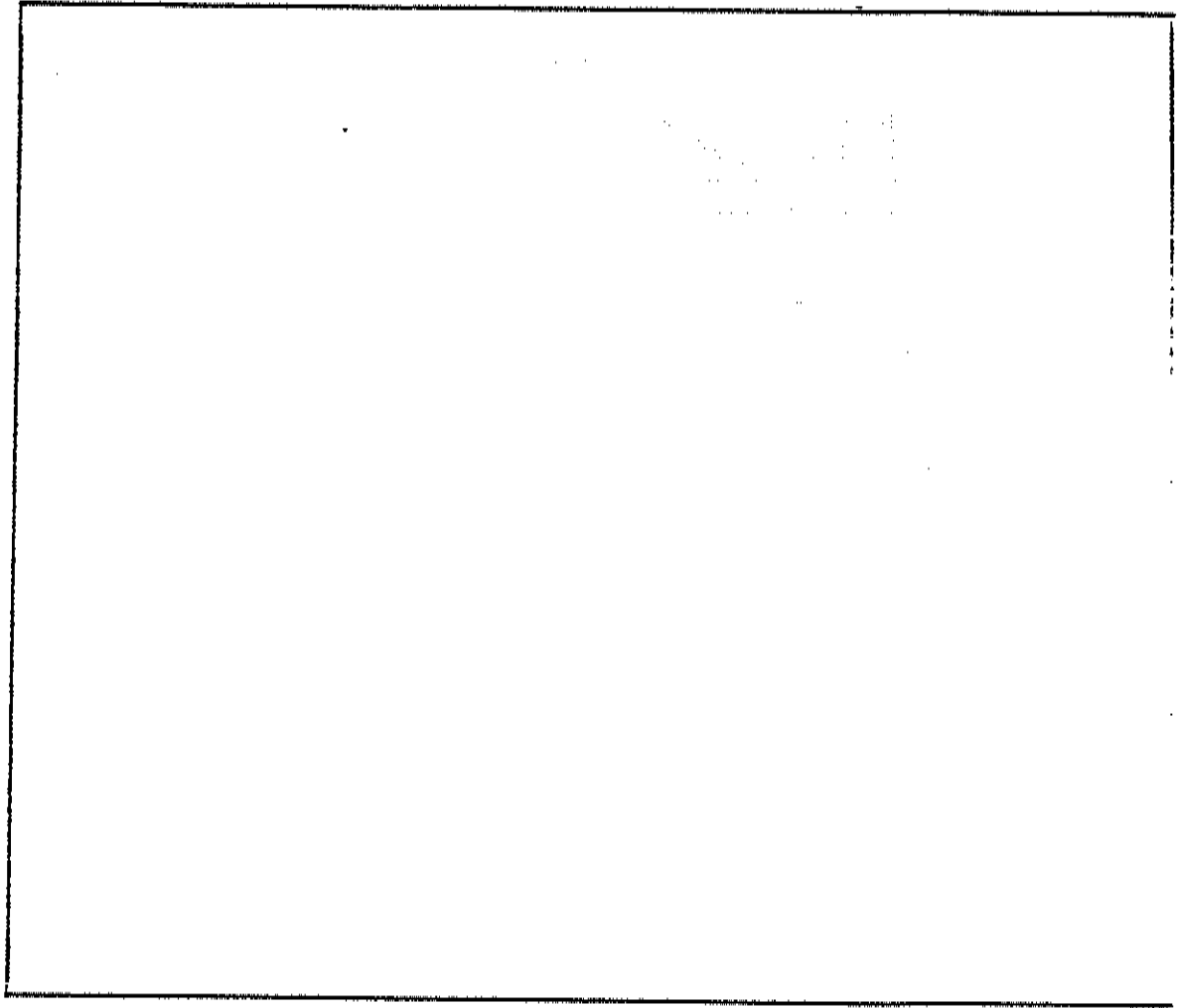
The complete instruction set is:

<u>Instruction</u>	<u>Format</u>	<u>Semantics</u>
Load A	$\begin{array}{ c c } \hline 11 & 0 \\ \hline 10 & 9 \\ \hline 00 & A \\ \hline \end{array}$	$AC \leftarrow M[A]$
Store A	$\begin{array}{ c c } \hline 01 & A \\ \hline \end{array}$	$M[A] \leftarrow AC$
Add A	$\begin{array}{ c c } \hline 10 & A \\ \hline \end{array}$	$AC \leftarrow AC + M[A]$
SKZ	$\begin{array}{ c c } \hline 11 & \text{---} \\ \hline \end{array}$	If $(AC=0)$ then skip next inst.

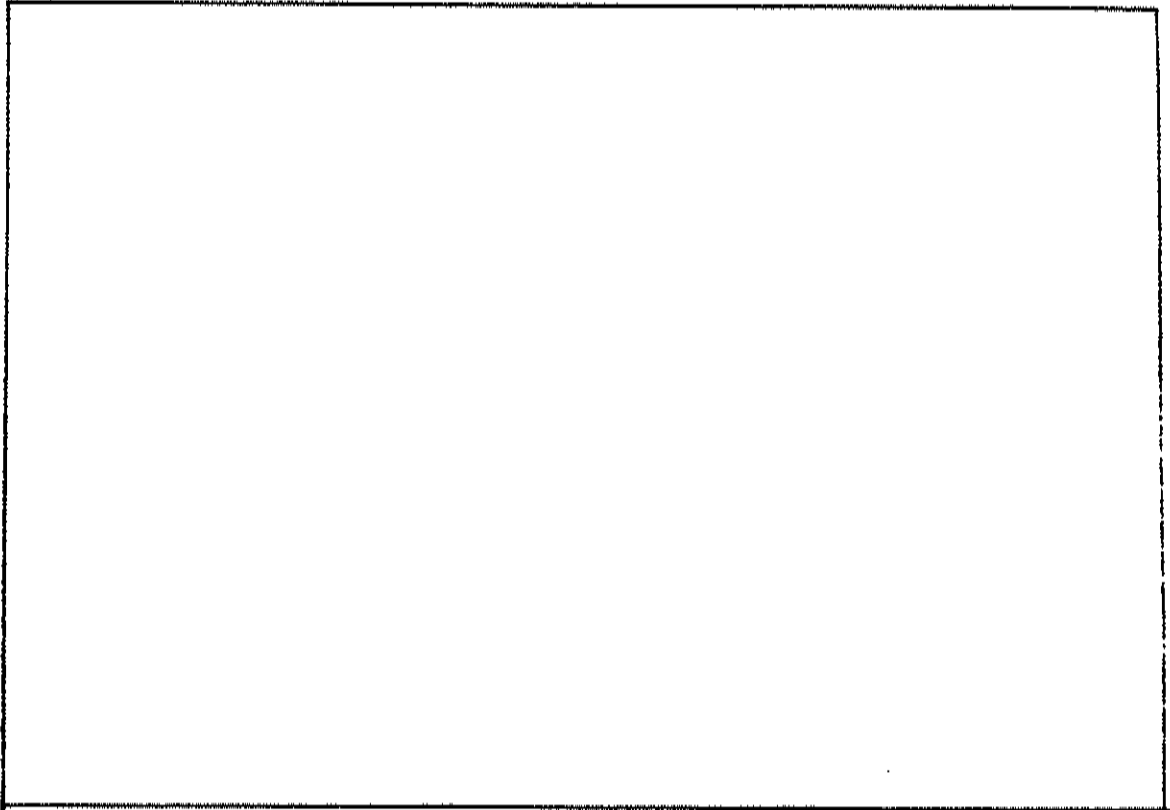
Note: Put all your answers only in the spaces provided.

Be neat!

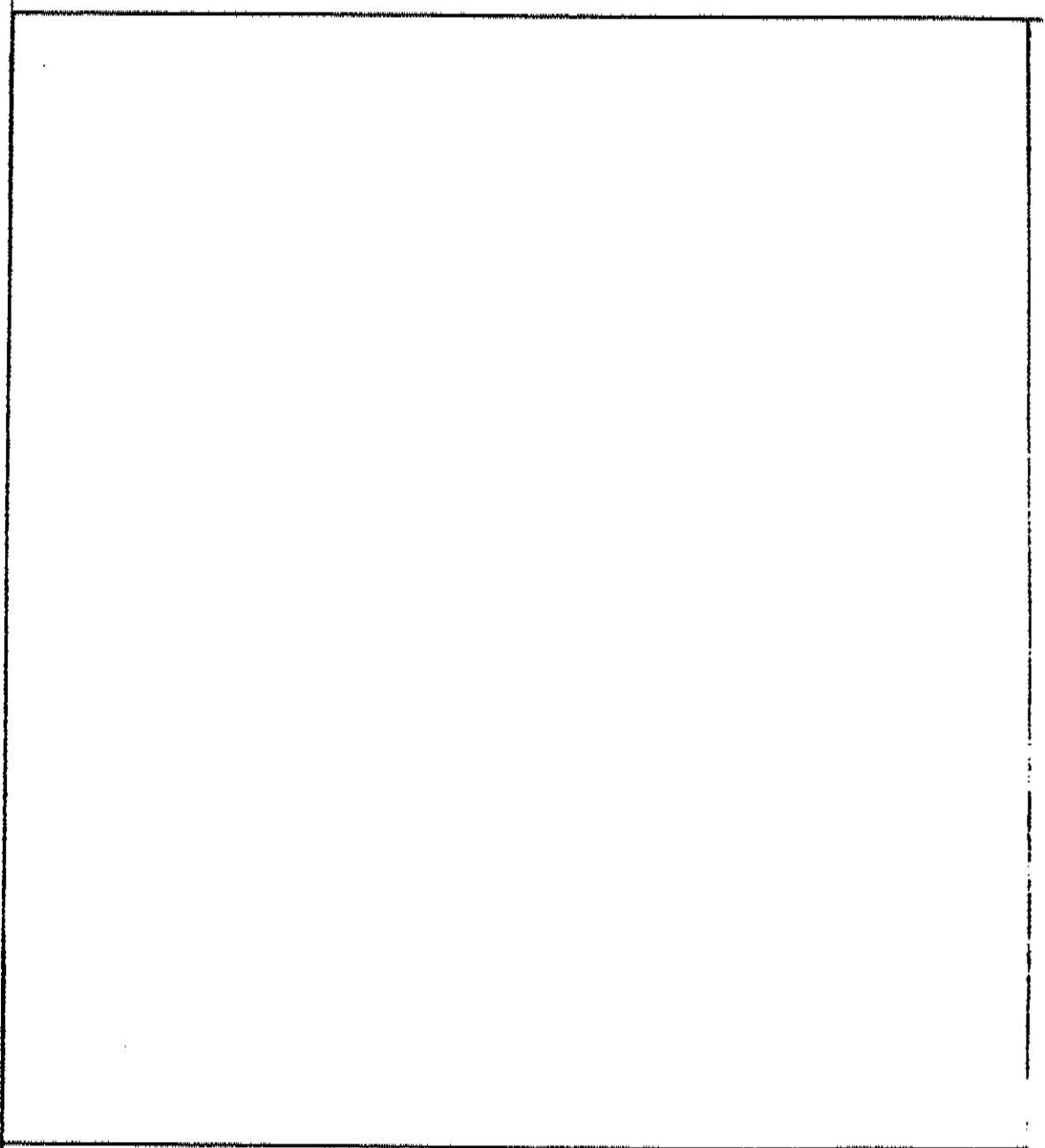
A. Describe the computer in ISP:



B. Draw a register diagram of the computer:



C. Draw a state diagram of the computer:

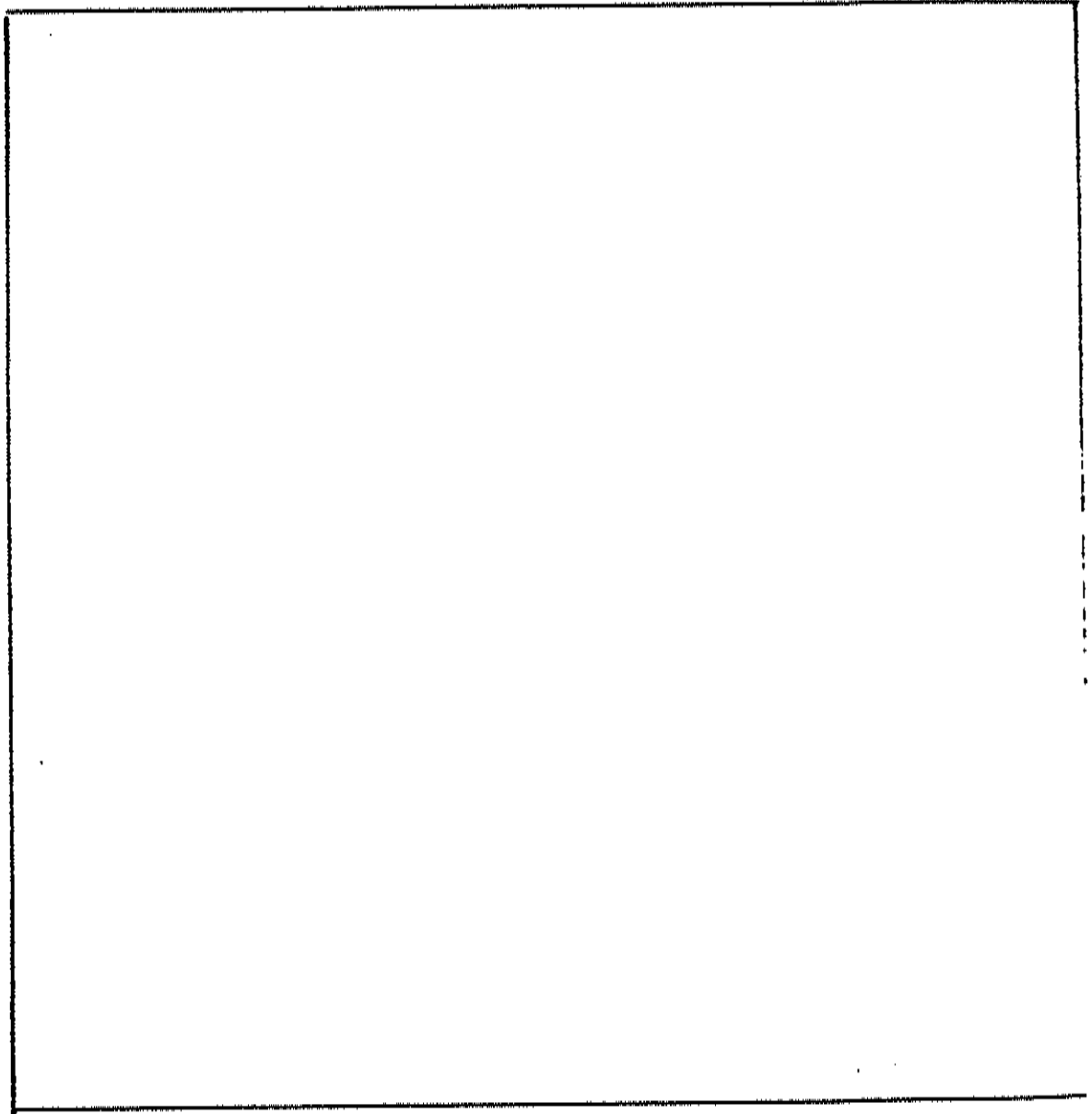
A large, empty rectangular box with a thin black border, intended for the student to draw a state diagram of a computer. The box is centered on the page and occupies most of the lower half of the page.

D. Draw a circuit of the ALU of the computer:

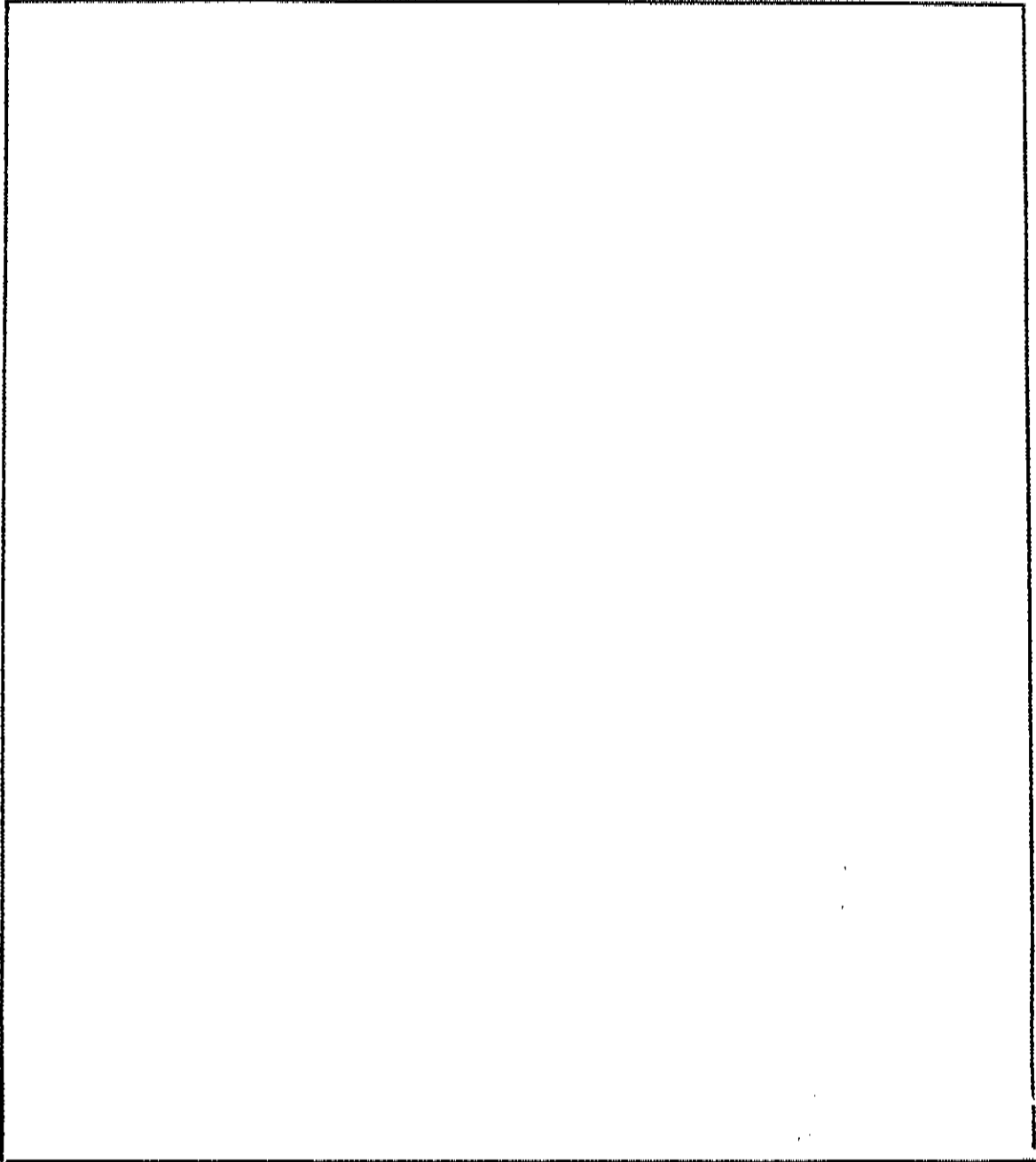
i) ALU in terms of 'bit-slices':

ii) Truth Table of a one-bit ALU slice:

iii) Karnaugh Map of a one-bit ALU Slice:

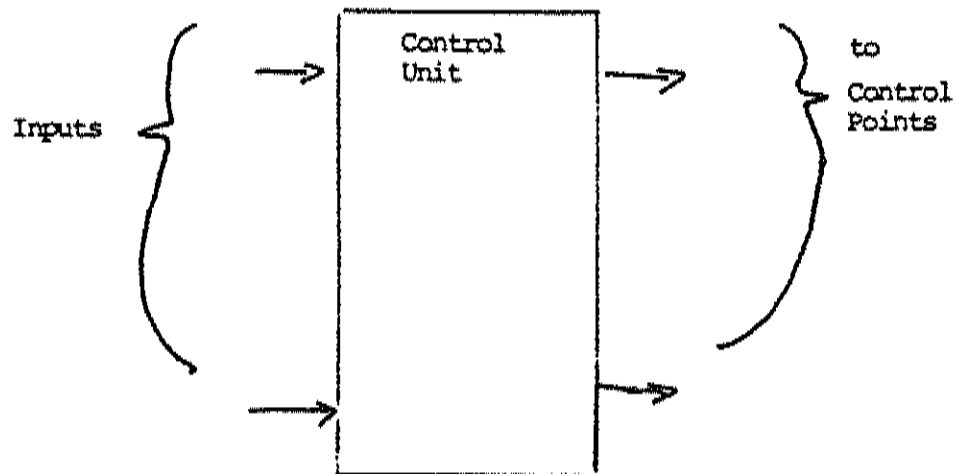


iv) NAND GATE (only) circuit of ALU Bit Slice:

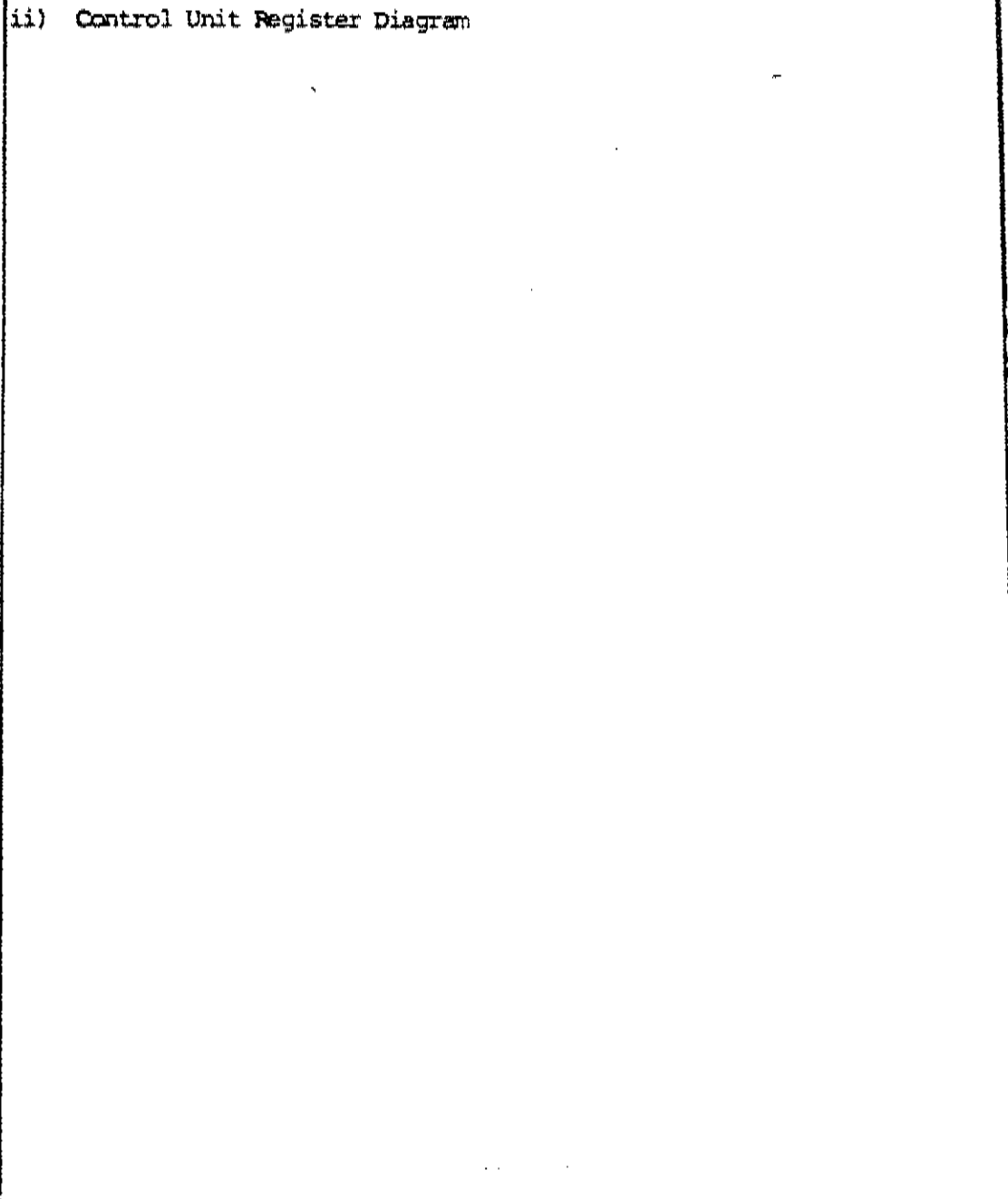


- E. Given your state diagram and register diagram, show a complete circuit diagram of your control unit. You must identify and design all aspects of the control including the identification of the control register and its implementation in NAND gates. Carefully organize your work and employ and show all truth tables, K-maps, etc. as needed. Be organized and neat!

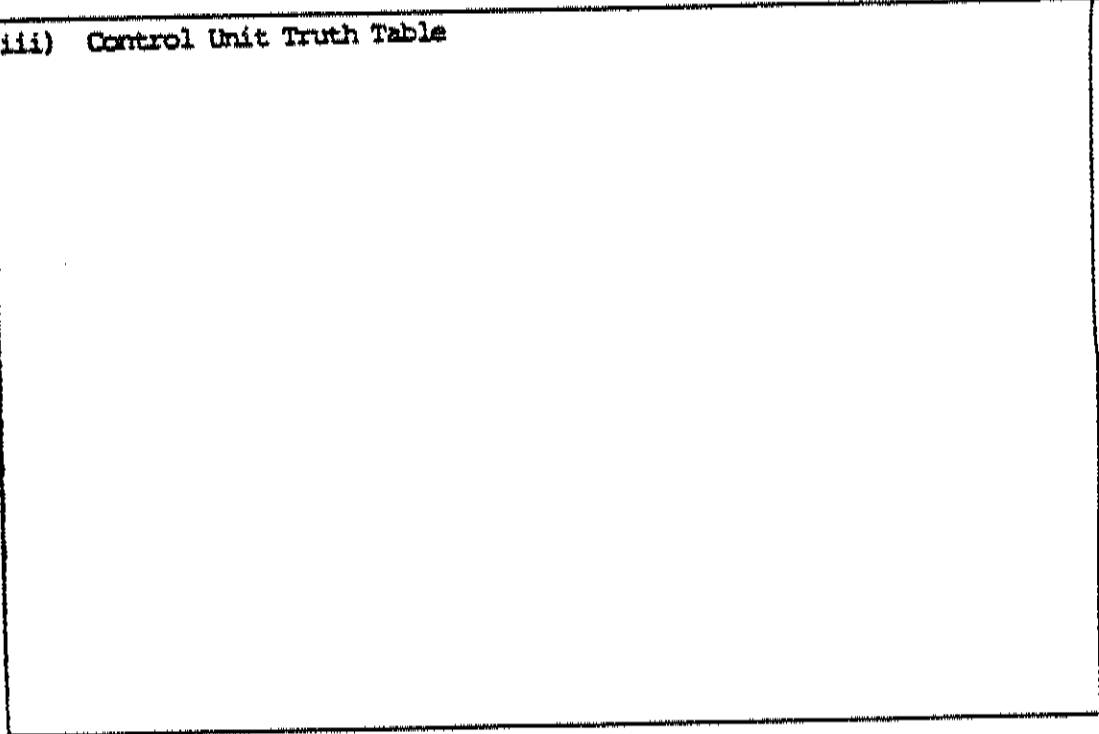
i) Label all the inputs and output to your control unit:



ii) Control Unit Register Diagram



(ii) Control Unit Truth Table



Hardware Core Exam: Fall 1986

University of California
HARDWARE PRELIM EXAM

Fall 1986

Your code number:

35

General Instructions:

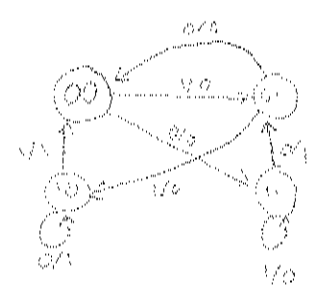
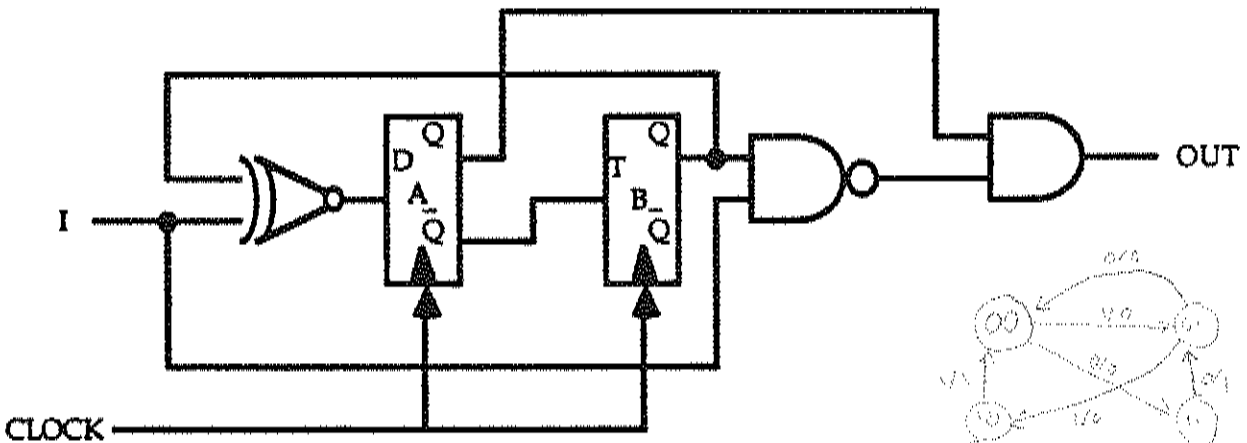
The exam lasts 3 hours = 180 minutes.
 Do all your work on these exam papers.
 Read the problem statements carefully.
 If something seems unclear, state your assumptions; you should not need to ask questions.
 The indicated points give you an idea how long a problem should take (minutes).

1. Draw the complete state diagram of the following Mealy machine comprising a D-flip flop and a Toggle flip flop.

Label the transition arrows with the input/output values in the following style (see also problem 4 for an example):

input / output_for_this_input

(15p)



$D = Q_B \oplus I$ $T = \bar{Q}_A$ $OUT = Q_A \cdot (\bar{Q}_B + I)$

Q_A	Q_B	I	Q_A^*	Q_B^*	OUT	D	T
0	0	0	1	1	0	1	1
0	0	1	0	1	0	1	1
0	1	0	0	0	0	0	1
0	1	1	1	0	0	0	1
1	0	0	0	0	0	0	1
1	0	1	0	0	0	0	1
1	1	0	0	1	0	0	0
1	1	1	1	1	1	0	0

038

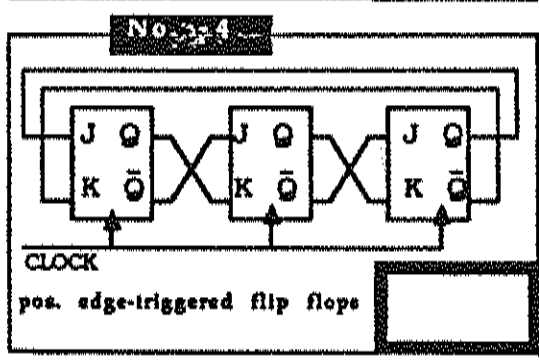
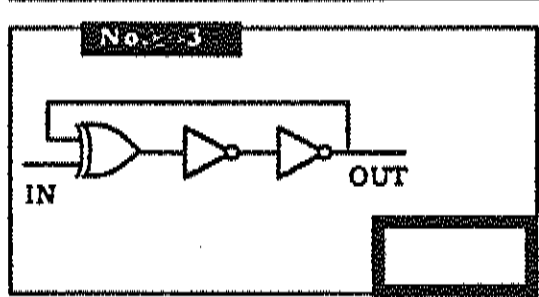
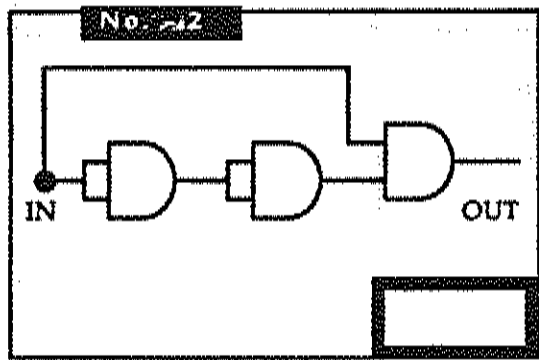
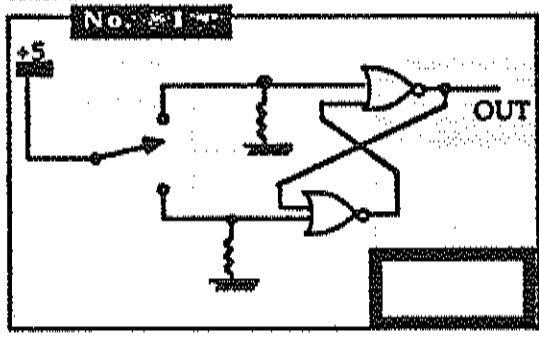
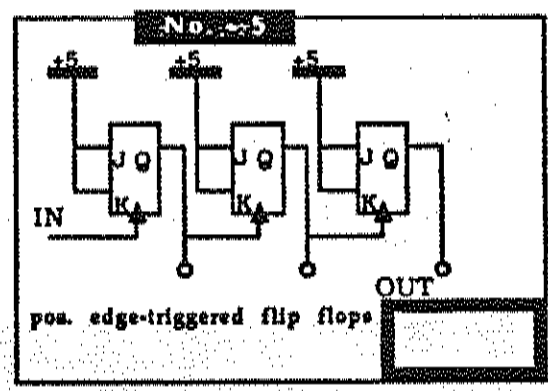
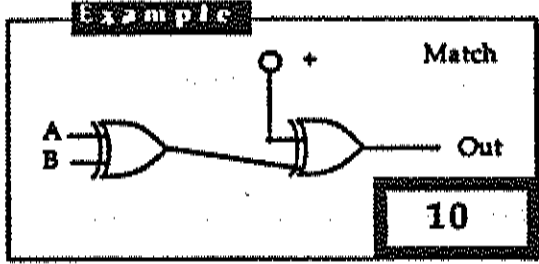
2. A Moore machine has six flip flops, four input pins and nine observable signal outputs. Consider the complete state diagram of this machine and fill in the proper numbers below:

(If you want to indicate that the correct answer should be "exactly xx", fill in "xx" in the MIN field and in the MAX field.)

(10p)

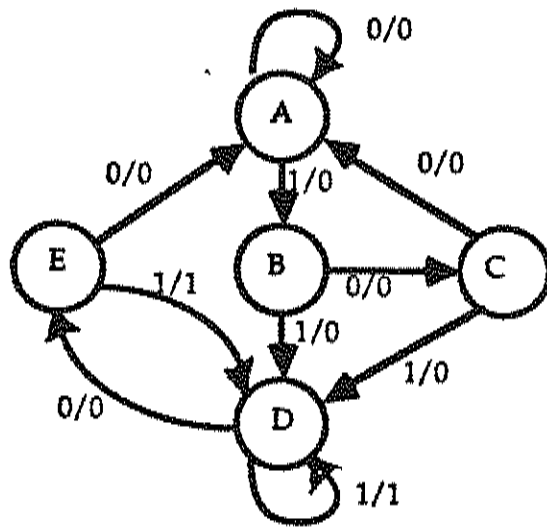
	MIN	MAX
The number of states of this machine:	2 ⁶	2 ⁶
The number of transition arrows starting from any particular state (counting multiple coinciding arrows with proper multiplicity):	4	4
The number of transition arrows ending up in a particular state:	0	2 ⁴
The number of different value patterns displayed on the output pins:	1	2 ⁶

3. Match each circuit below with the one sentence that best describes it. (15p)



1. Device to DETECT and REMEMBER the occurrence of a single input pulse.
2. Up counter
3. Down counter
4. Serial shift register
5. Parallel shift register
6. PULSE SHAPER: Outputs a single short pulse in response to the rising edge of an input pulse.
7. PULSE SHAPER: Outputs a single short pulse in response to the falling edge of an input pulse.
8. EDGE DETECTOR: Outputs a single short pulse in response to a change in the output (0 to 1) or (1 to 0).
9. Debounced switch
10. Exclusive - NOR
11. Clock or Oscillator
12. Clock with ON/OFF switch
13. Ring counter
14. Twisted (Moebius) Ring Counter
15. None of the above

4. Given the following reduced FSM,



(a) Select a good state encoding (include a brief description of your rationale!)

(10)

$A = 0001$
 $E = 0111$
 $C = 0101$
 $B = 1111$
 $D = 1011$

$z_0 = 0001 + 1001 + 1011 + 1010 + 0111 + 0101 + 1101$
 $z_1 = 1001 + 0011 + 1011 + 1010 + 1111 + 1101$
 $z_2 = 1011 + 1010 + 0111 + 1111 + 1101$
 $z_3 = 1111 + 1101$

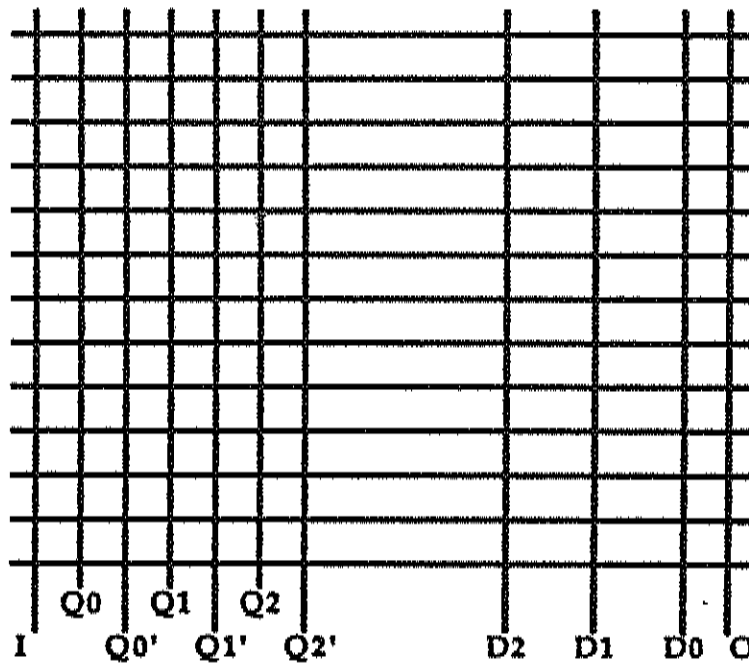
	Q_3	Q_2	Q_1	Q_0	I	Q_3	Q_2	Q_1	Q_0	z_3
0	0	0	0	1	0	0	0	1	0	0
1	0	1	1	1	1	0	1	1	0	0
2	0	1	0	1	0	0	1	0	0	0
3	1	1	1	1	1	1	1	1	0	0
4	0	0	1	0	0	0	0	0	0	0
5	0	0	1	1	1	1	1	1	0	0
6	0	1	1	0	1	1	0	1	0	0
7	1	1	1	1	1	1	1	1	1	0
8	0	1	0	0	0	0	0	1	0	0
9	1	1	1	1	1	1	1	1	1	0

(b) Develop the Boolean equations for the next state and output, assuming the state register is implemented with D flipflops.

(5)

(c) Program the following PLA template to implement the controller:

(5)



write down the min-terms realized by each row

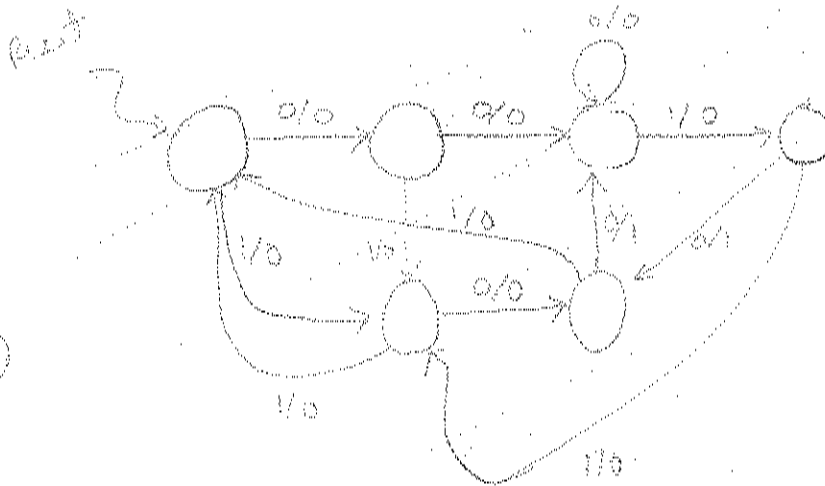
5. Design a clocked sequential network that outputs $Z = 1$ for every input sequence ending in 0010 or 100,

e.g., $X = 110010010100101$

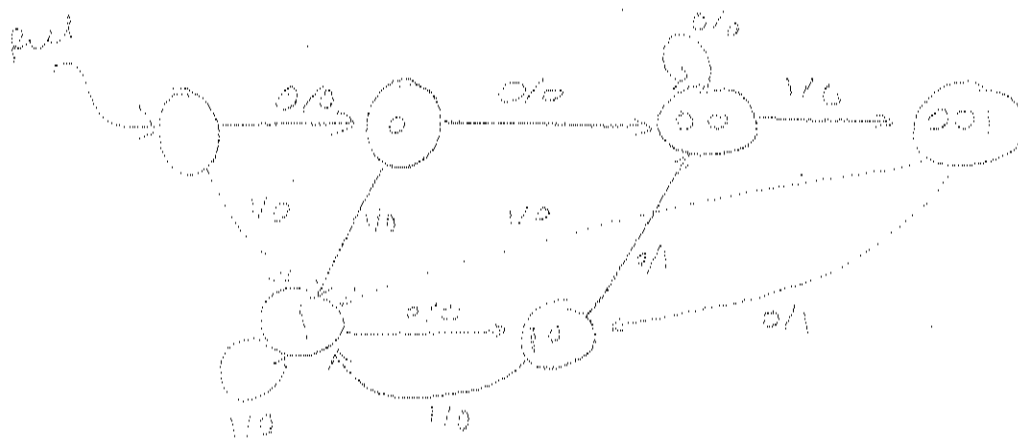
$Z = 000101101001010$

Draw a MINIMUM state state diagram for this recognizer assuming you are implementing it as a MEALY MACHINE. You may also assume that the inputs change with the clock edge.

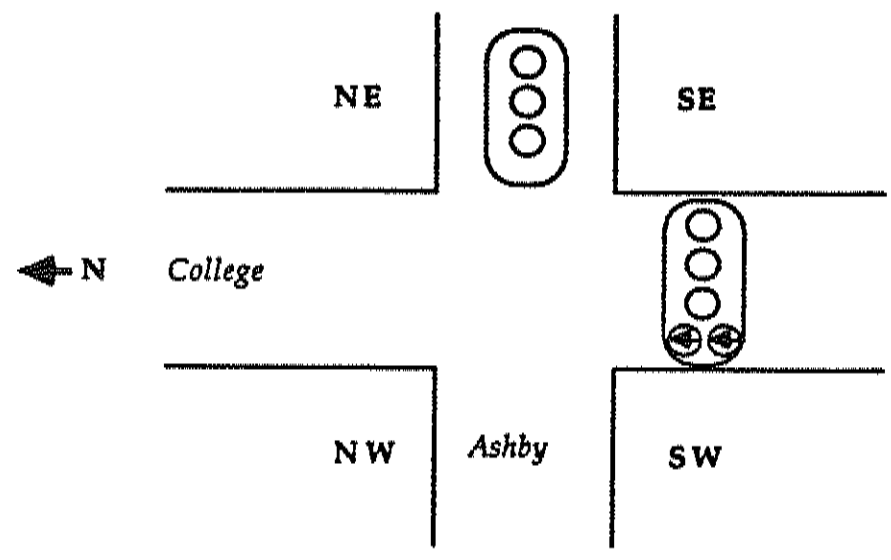
(15 pts).



Look at this more carefully



- 6. Two two-way streets meet at an intersection which is controlled by a traffic light. In the east, west, and south directions, the traffic lights are conventionally red, yellow, and green, but the lights facing north are augmented with green and yellow left turn arrows that may be OFF or illuminated.



You may assume:

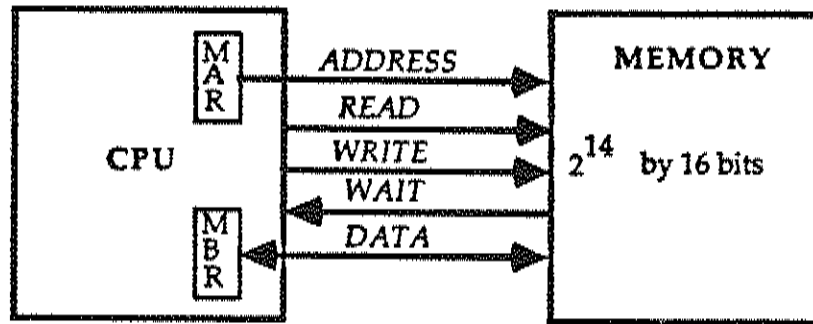
- (1) You have as many programmable timers as you need. These can be loaded with a time constant and assert their output when they count down to zero.
- (2) The north facing lights cycle red (60 seconds) - green arrow (20 seconds) - yellow arrow (10 seconds) - green (45 seconds) - yellow (15 seconds) - red (60 seconds).

Draw the state diagram, and explicitly list all inputs and output control signals needed for the complete traffic light system.

(25 pts).

7. Consider the following specification of a very simple CPU:

Memory Interface:



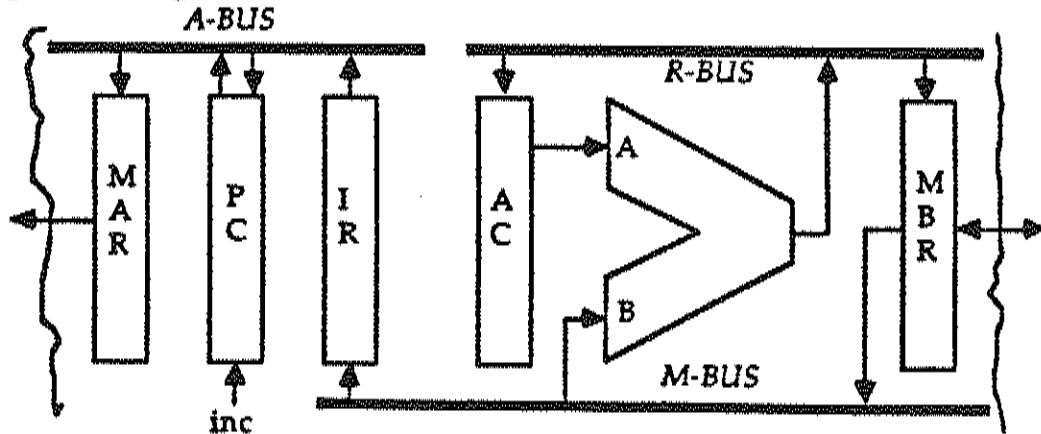
Memory will assert WAIT immediately after the CPU has asserted READ or WRITE. Data will be valid on a READ or can be removed on a WRITE only after the WAIT signal is no longer asserted by the memory.

Instruction Format:



00	LOAD <ADDRESS>	AC := MEM[<ADDR>]
01	STORE <ADDRESS>	MEM[<ADDR>] := AC
10	ADD <ADDRESS>	AC := AC + MEM[<ADDR>]
11	BRANCH NEGATIVE <ADDRESS>	IF AC < 15 = 1 THEN PC := <ADDR>

Register Diagram:

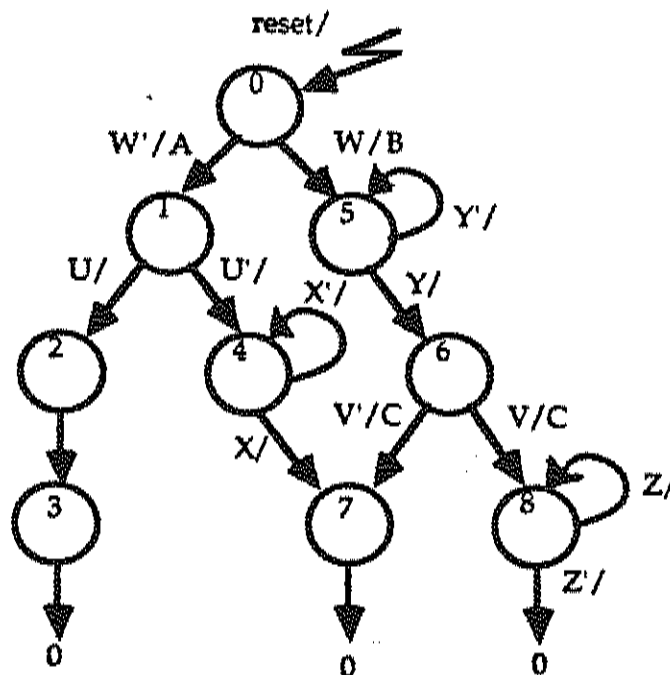


The basic instruction sequencing is instruction fetch, instruction decode, and instruction execution.

List the necessary register transfer microoperations required within the datapath to implement the specified instruction set.

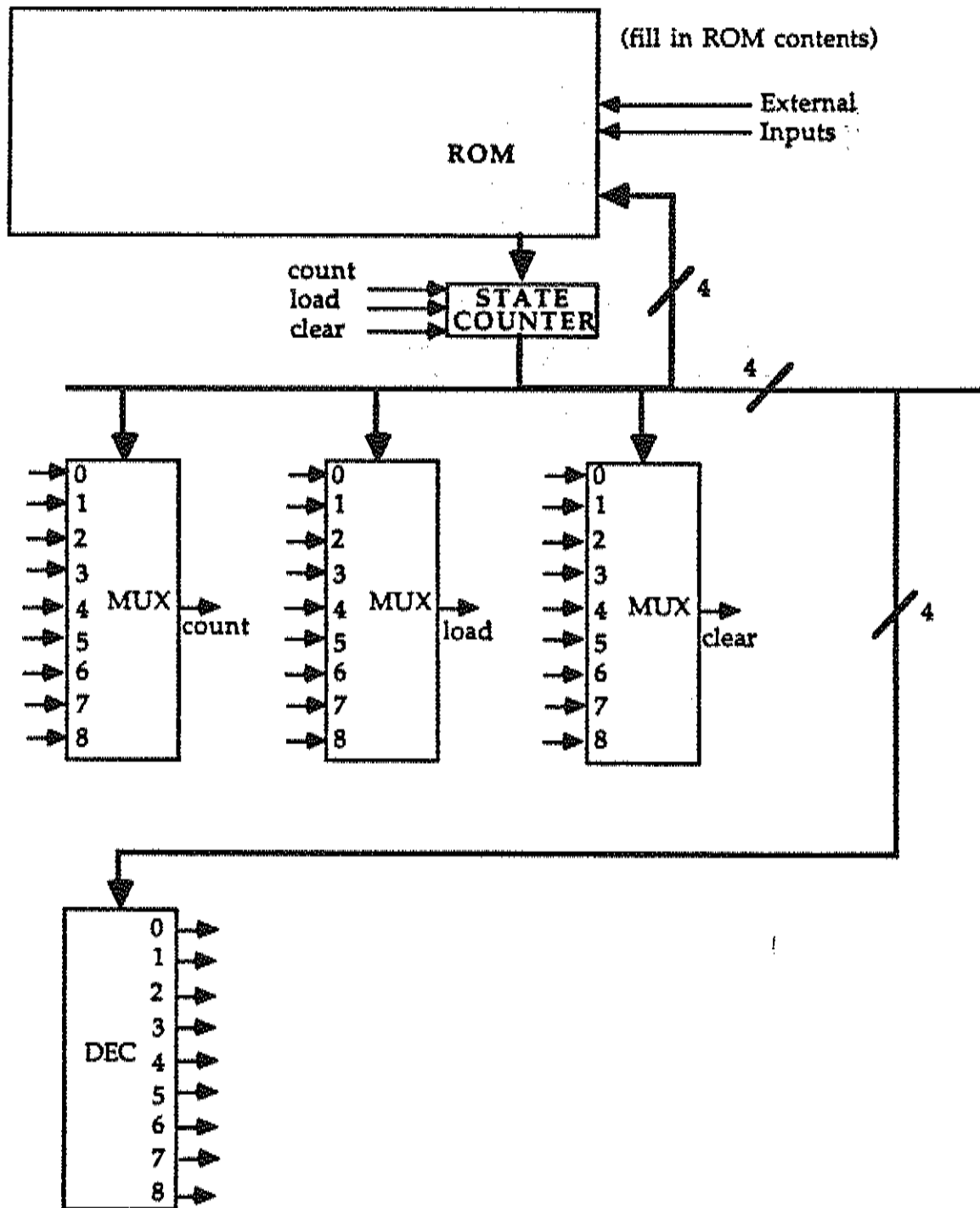
(20 pts).

8. Given the following state diagram:



Implement this controller using a hybrid jump counter approach. A jump counter implements the state register with a counter that can (i) count up, (ii) be loaded from a ROM, or (iii) be reset. You may assume that LOAD/CLEAR is synchronous to the clock and takes priority over the count signal.

Show the contents of your ROM and how you have wired up the COUNT, LOAD, and CLEAR signals on the diagram on the next page. Also show how the output signals A, B, and C are generated. (15 pts).



9. The clock cycle (i.e., the microcycle) of a CPU datapath is usually constrained by the delays incurred in one of three critical paths. Your job in this problem is to identify (for each path) the critical sequence of events that must occur within one cycle. Also, for each path, discuss one element in the path with respect to minimizing its effect on the cycle time.

Path 1. (HINT: Use of virtual memory is often a factor) (5)

Path 2. (HINT: The size of the scratchpad register file is a factor) (5)

Path 3. (HINT: If the machine has few instructions, this is NOT a factor) (5)

10. Draw a "register-level" diagram of a cache memory system to support a high speed CPU. Show sufficient details so that it can be seen how each feature specified below can operate. You may assume that your components are random access static memories modules, bus multiplexors, tri-state bus drivers, ALUs, registers, and ordinary logic gates and flip-flops. Show explicitly the number of bits used in each block and justify your choice. (30 pts).

FEATURE/PARAMETER SPECIFICATION:

- Two-way set associative
- Write back
- FIFO replacement
- Two words/line
- 4K total words of data in the cache
- 16 bits/word
- 16 bit address bus (64K address space)

164/10

Number: 9

General Instructions:

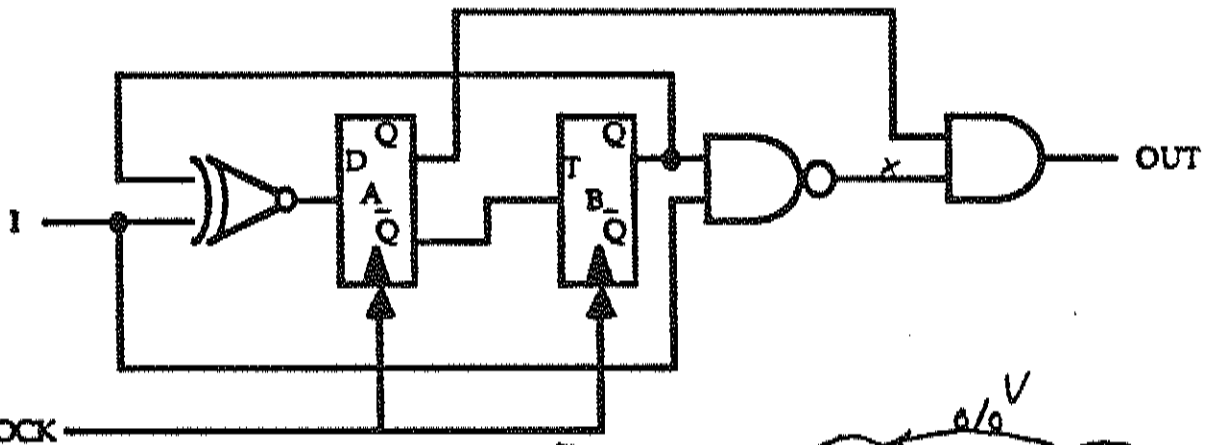
The exam lasts 3 hours = 180 minutes.
Do all your work on these exam papers.
Read the problem statements carefully.
If something seems unclear, state your assumptions; you should not need to ask questions.
The indicated points give you an idea how long a problem should take (minutes).

1. Draw the complete state diagram of the following Mealy machine comprising a D-flip flop and a Toggle flip flop.

Label the transition arrows with the input/output values in the following style (see also problem 4 for an example):

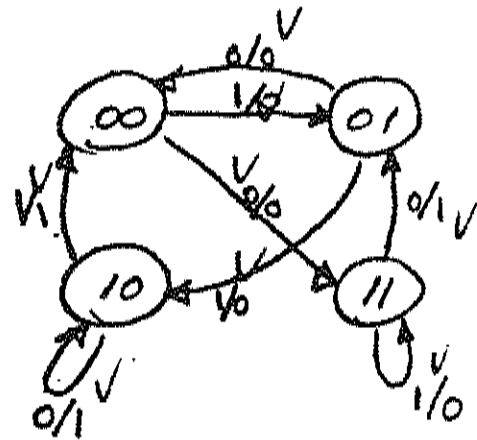
(15p)

input / output_for_this_input



I	Q _A	Q _B	D _A	T _B	X	OUT
0	0	0	1	1	1	0
0	0	1	0	1	1	0
0	1	0	0	0	1	0
0	1	1	0	0	1	0
1	0	0	0	1	1	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	1	0	0	0

$D_A = \overline{B}Q_B$
 $T_B = \overline{Q_A}$
 $X = Q_B I$
 $OUT = XQ_A$



15

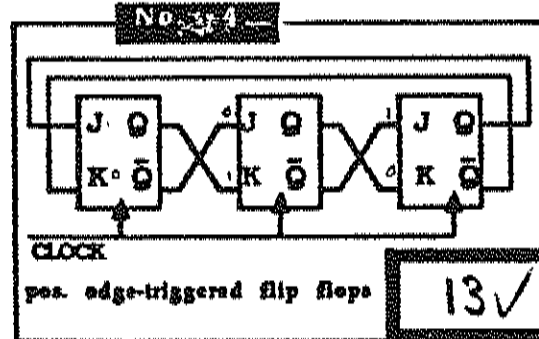
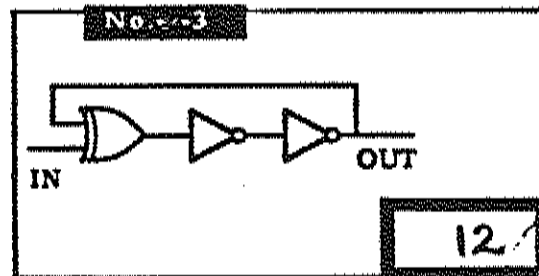
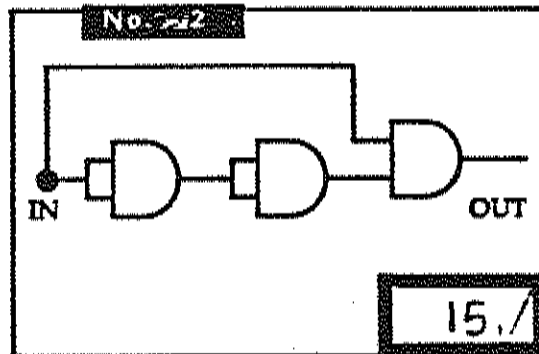
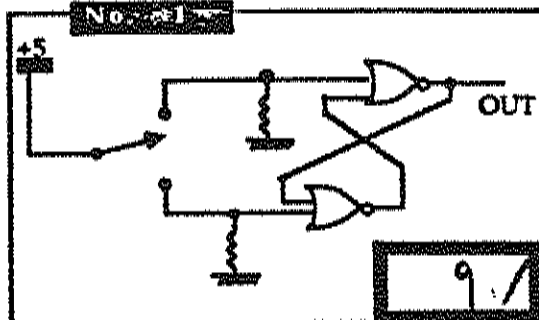
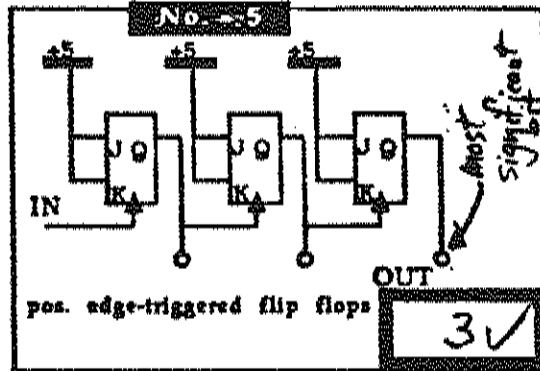
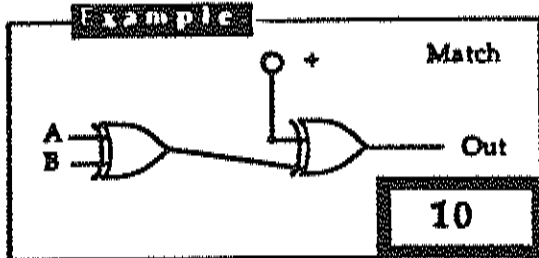
2. A Moore machine has six flip flops, four input pins and nine observable signal outputs. Consider the complete state diagram of this machine and fill in the proper numbers below:

(If you want to indicate that the correct answer should be "exactly xx", fill in "xx" in the MIN field and in the MAX field.)

(10p) 30

	MIN	MAX
The number of states of this machine:	1	2^6 ✓
The number of transition arrows starting from any particular state (counting multiple coinciding arrows with proper multiplicity):	2^4 ✓	2^4 ✓
The number of transition arrows ending up in a particular state:	0 ✓	$2^6 \cdot 2^4$ ✓
The number of different value patterns displayed on the output pins:	1 ✓	2^6 ✓

3. Match each circuit below with the one sentence that best describes it.

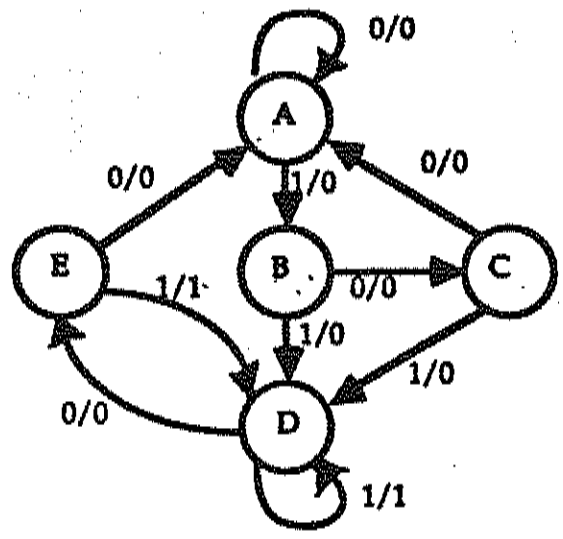


1. Device to DETECT and REMEMBER the occurrence of a single input pulse.
2. Up counter
3. Down counter
4. Serial shift register
5. Parallel shift register
6. PULSE SHAPER: Outputs a single short pulse in response to the rising edge of an input pulse.
7. PULSE SHAPER: Outputs a single short pulse in response to the falling edge of an input pulse.
8. EDGE DETECTOR: Outputs a single short pulse in response to a change in the output (0 to 1) or (1 to 0).
9. Debounced switch
10. Exclusive - NOR
11. Clock or Oscillator
12. Clock with ON/OFF switch
13. Ring counter
14. Twisted (Moebius) Ring Counter
15. None of the above

34

053 Bravo! 15

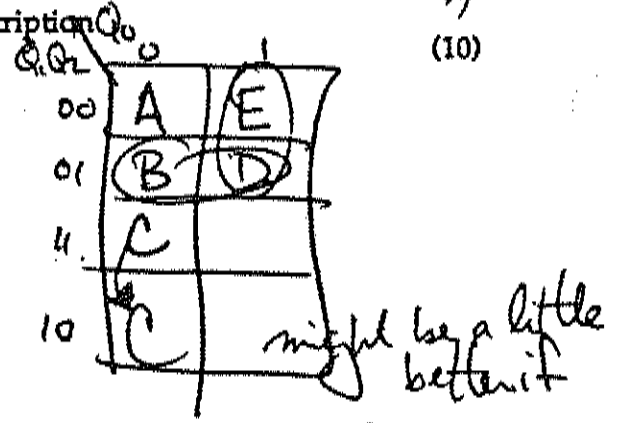
4. Given the following reduced FSM,



(a) Select a good state encoding (include a brief description of your rationale!)

	NS		OUT		encoding
	0	1	0	1	
A	A	B	0	0	0 00
B	C	D	0	0	0 01
C	A	D	0	0	0 11
D	E	D	0	1	1 01
E	A	D	0	1	1 00

$D_2 D_1 D_0$



5 states \Rightarrow 3 bits.

- (1) use most sig bit to distinguish outputs (ie. give states with same output the same sig. bit) ✓
- (2) Now try to make states that are connected by transitions differ by only one bit. ✓

Only $C \rightarrow D + C \rightarrow A$
 violate (2) — they differ by two

$C = 010$
 then differs by 1 from A

(b) Develop the Boolean equations for the next state and output, assuming the state register is implemented with D flipflops.

(5)

$D_2, D_1, D_0 \equiv \text{state}_{next}$

D_2

	Q_1, Q_0			
	00	01	11	10
I, Q_2	0	0	0	X
	0	1	X	X
	1	1	X	X
	1	0	1	X

$D_2 = I Q_2 + I Q_0 + Q_2 Q_0$

D_0

	0	1	0	X
	0	0	X	X
	1	1	X	X
	1	0	1	X

$D_0 = I + \overline{Q_2} \overline{Q_1} Q_0$

D_1

	0	0	0	X
	0	0	X	X
	0	0	X	X
	0	0	0	X

$D_1 = \overline{I} \overline{Q_2} \overline{Q_1} Q_0$

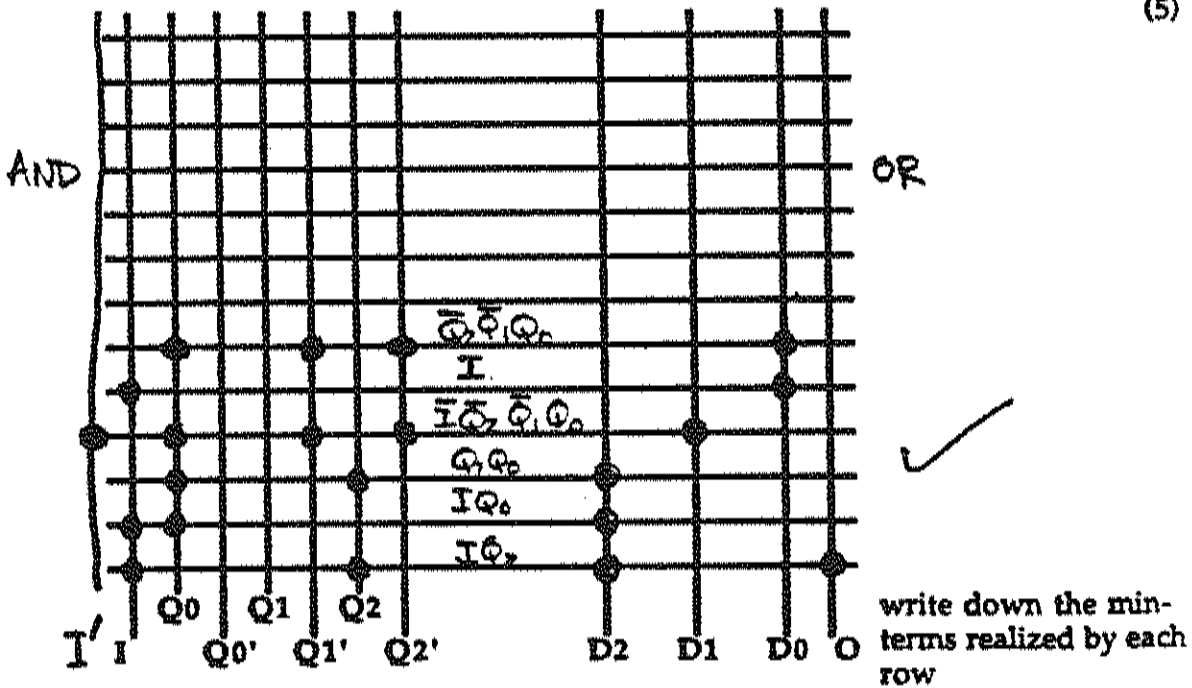
OUT

	0	0	0	X
	0	0	X	X
	1	1	X	X
	0	0	0	X

OUT = $I Q_2$

(c) Program the following PLA template to implement the controller:

(5)



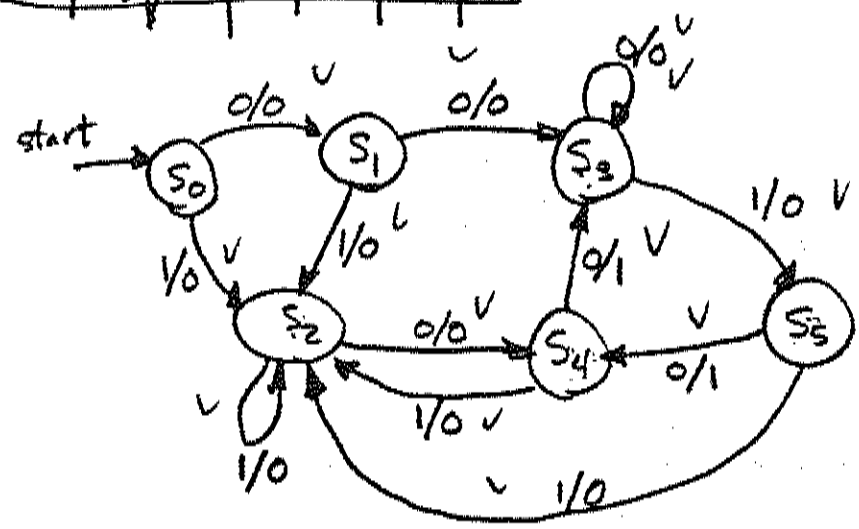
5. Design a clocked sequential network that outputs $Z = 1$ for every input sequence ending in 0010 or 100,

eg., $X = 110010010100101$
 $Z = 000101101001010$

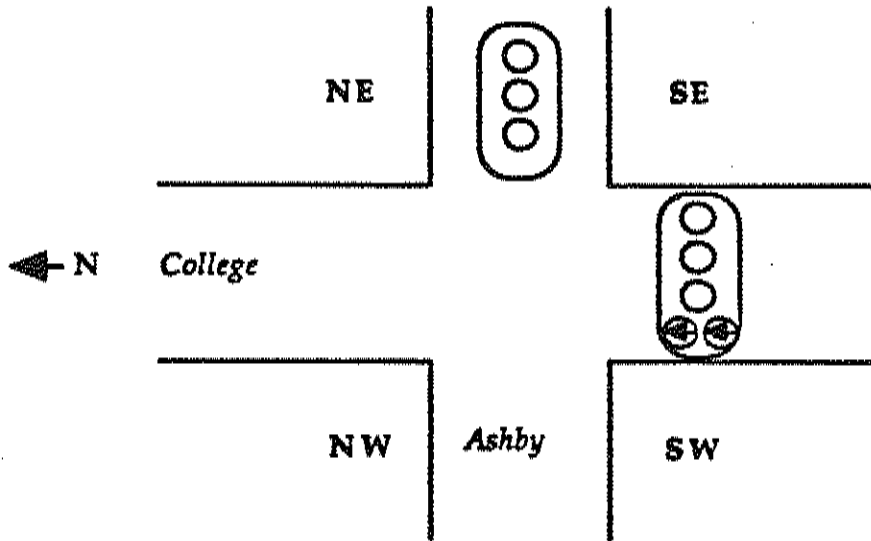
Draw a MINIMUM state state diagram for this recognizer assuming you are implementing it as a MEALY MACHINE. You may also assume that the inputs change with the clock edge.

20
(15 pts).

current	NS	0	1	OUT	0	1
—	S_0	S_1	S_2	0	0	
0	S_1	S_3	S_2	0	0	
1	S_2	S_4	S_2	0	0	
00	S_3	S_3	S_5	0	0	
10	S_4	S_3	S_2	1	0	
001	S_5	S_4	S_2	1	0	



6. Two two-way streets meet at an intersection which is controlled by a traffic light. In the east, west, and south directions, the traffic lights are conventionally red, yellow, and green, but the lights facing north are augmented with green and yellow left turn arrows that may be OFF or illuminated.



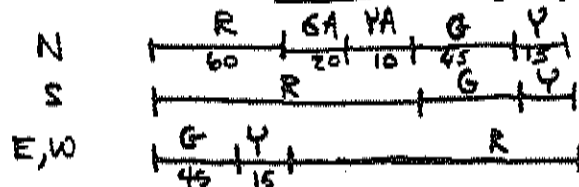
You may assume:

(1) You have as many programmable timers as you need. These can be loaded with a time constant and assert their output when they count down to zero.

(2) The north facing lights cycle red (60 seconds) - green arrow (20 seconds) - yellow arrow (10 seconds) - green (45 seconds) - yellow (15 seconds) - red (60 seconds).



Draw the state diagram, and explicitly list all inputs and output control signals needed for the complete traffic light system.



35
(25 pts).

↑ seconds E,W direction

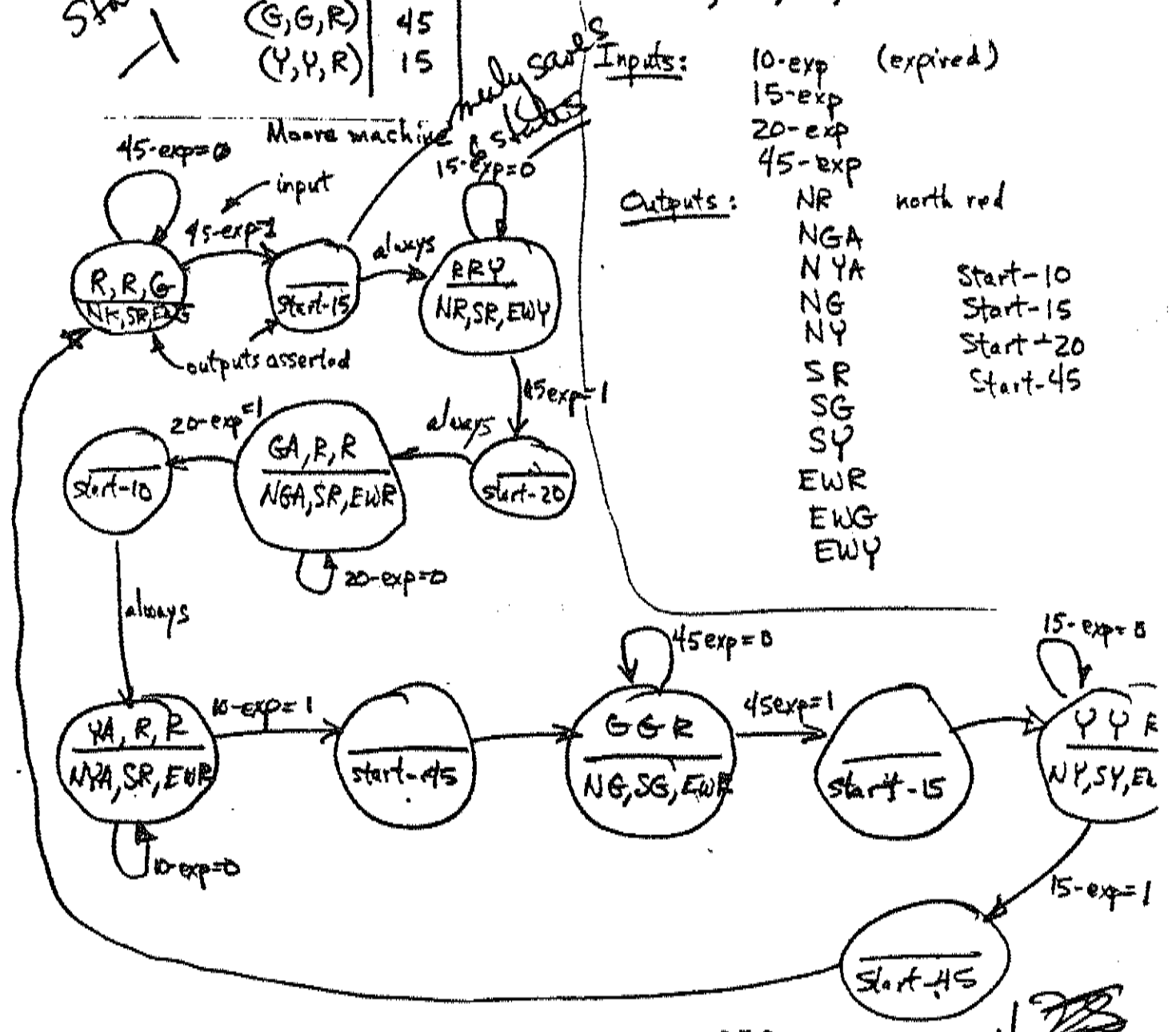
Let the state be given by the triplet (N,S,EW), the colors of the N, S, and EW lights.

state	time
(R,R,G)	45
(R,R,Y)	15
(G,R,R)	20
(Y,R,R)	10
(G,G,R)	45
(Y,Y,R)	15

Start?

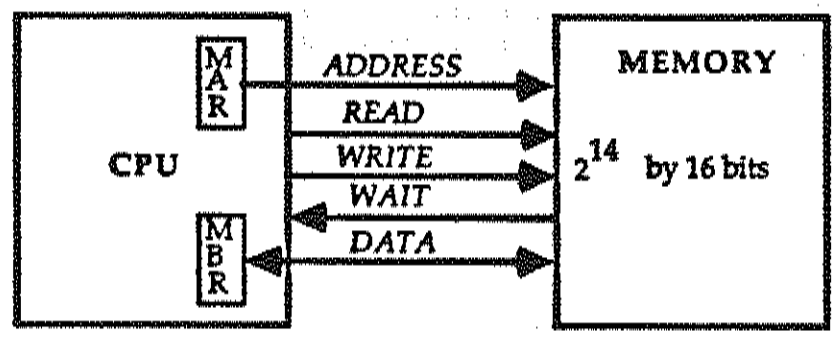
Assume that we have 10, 15, 20, and 45 timers.

- Inputs: 10-exp (expired)
 15-exp
 20-exp
 45-exp
- Outputs: NR north red
 NGA
 NYA
 NG
 NY
 SR
 SG
 SY
 EWR
 EWG
 EWY
- Start-10
 Start-15
 Start-20
 Start-45



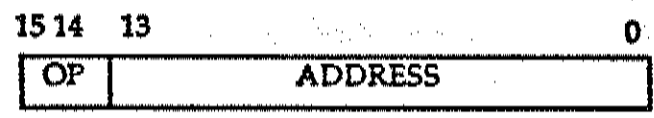
7. Consider the following specification of a very simple CPU:

Memory Interface:



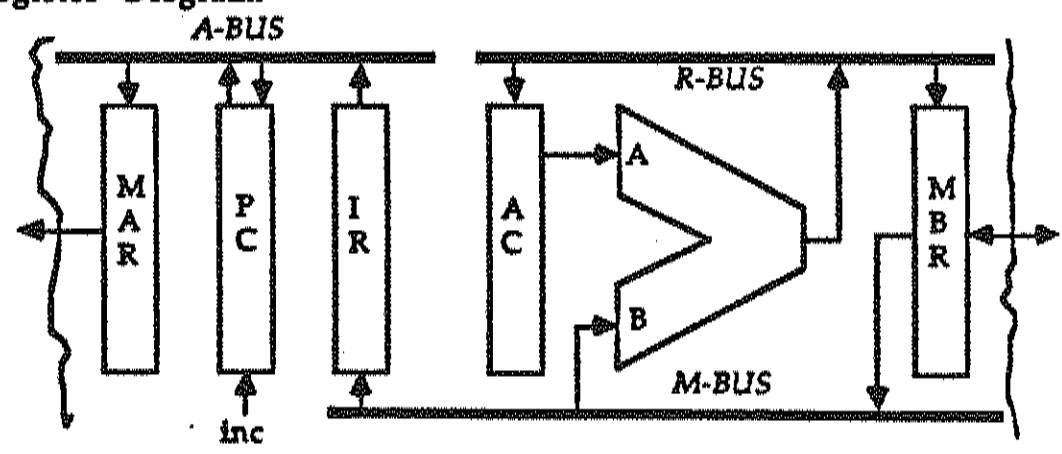
Memory will assert WAIT immediately after the CPU has asserted READ or WRITE. Data will be valid on a READ or can be removed on a WRITE only after the WAIT signal is no longer asserted by the memory.

Instruction Format:



- | | | |
|----|---------------------------|------------------------------------|
| 00 | LOAD <ADDRESS> | AC := MEM[<ADDR>] |
| 01 | STORE <ADDRESS> | MEM[<ADDR>] := AC |
| 10 | ADD <ADDRESS> | AC := AC + MEM[<ADDR>] |
| 11 | BRANCH NEGATIVE <ADDRESS> | IF AC<15> = 1
THEN PC := <ADDR> |

Register Diagram:



The basic instruction sequencing is instruction fetch, instruction decode, and instruction execution.

List the necessary register transfer microoperations required within the datapath to implement the specified instruction set.

05
(20 pts).

Fetch

- 1. PC → A bus
A bus → MAR ✓
- 2. MAR → Address bus
inc PC ✓
Mem read
Data bus → MBR
- remain in (2) until WAIT = 0

Decode

- 3. MBR → IR (M-bus)
- branch to next state based on MBR[15:14]

LOAD

- 4. IR<13:0> → MAR (A-bus)
- 5. MAR → Address bus
Mem read
Data bus → MBR
- remain in (5) until WAIT = 0
- 6. MBR → ALU-B (M-bus)
PASS-B
ALU → AC ✓ (R-bus)
- jump to 1

STORE

- 7. IR<13:0> → MAR (A bus)
AC → ALU-A
PASS A
ALU → MBR (R-bus)
- 8. MBR → data bus
MAR → addr bus
Mem write
remain in (8) until WAIT = 0
jump to 1.

ADD

- 9. IR<13:0> → MAR (A bus)
- 10. MAR → Addr bus
Mem read
Data bus → MBR
- remain in (10) until WAIT = 0
- 11. MBR → ALU-B
AC → ALU-A
ALU (+)
ALU → AC
jump to 1.

(OVER)

20

Branch

12.

if $AC<15> = 0$ then jump to 1.

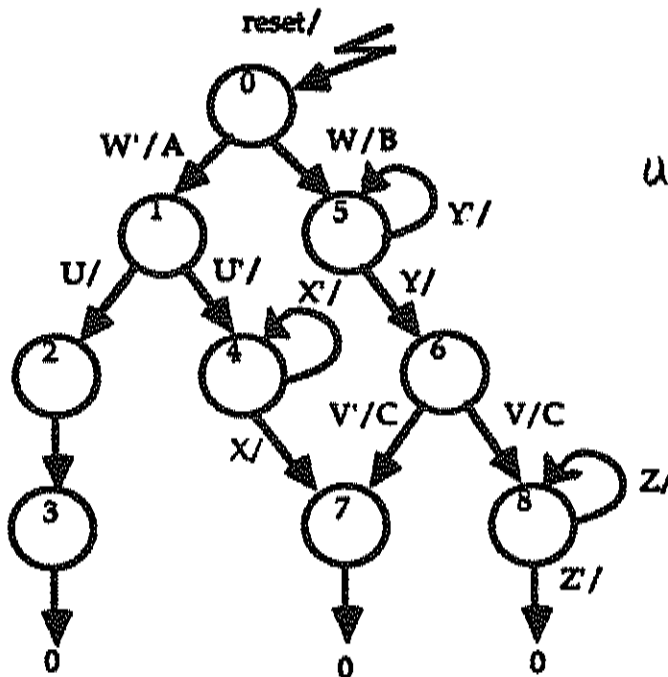
if $AC<15> = 1$ then jump to 13.

13.

$IR<13:0> \rightarrow PC$ (Abus)

jump to 1.

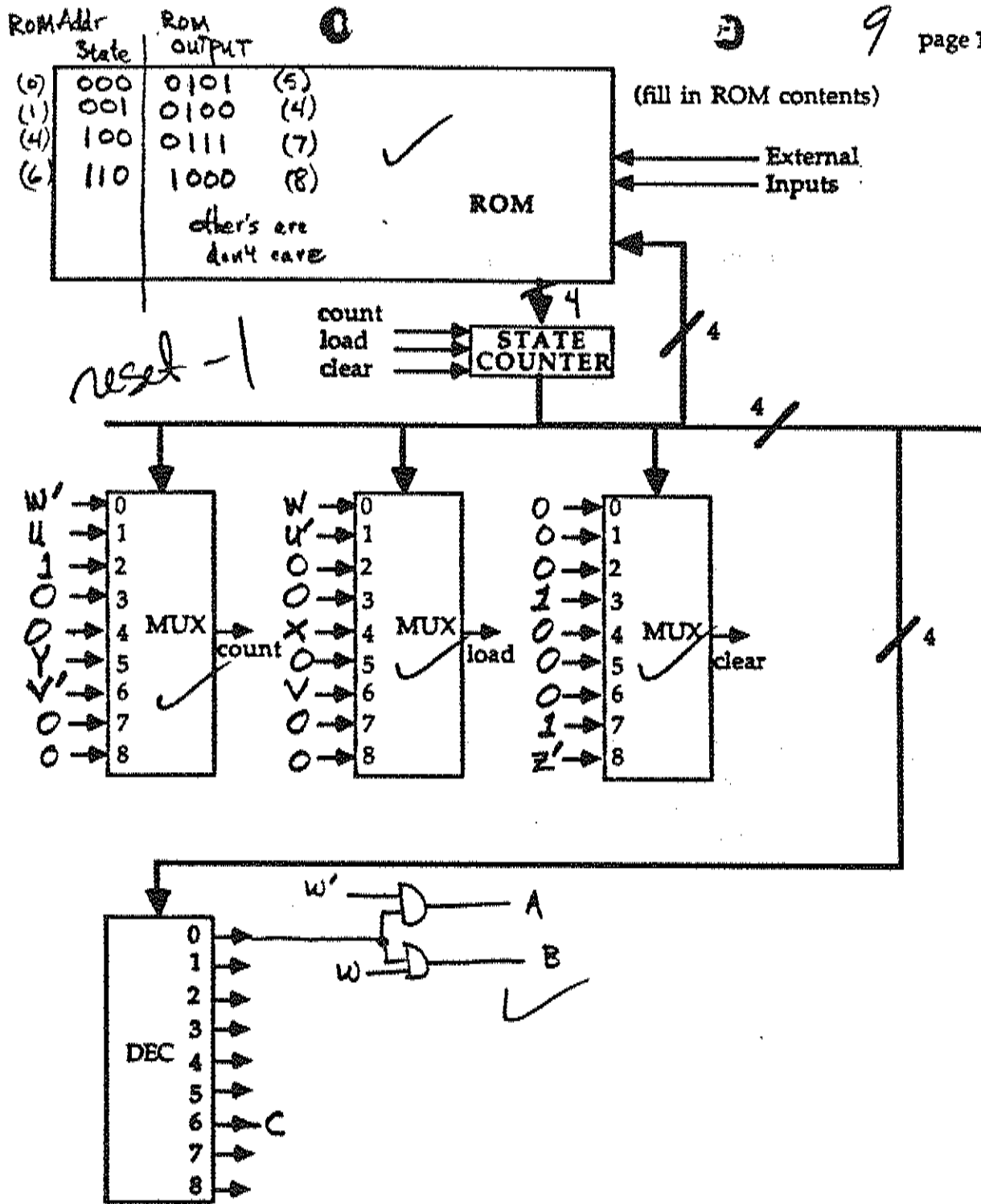
8. Given the following state diagram:



u v w x y z

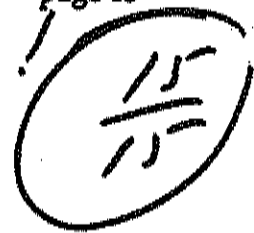
Implement this controller using a hybrid jump counter approach. A jump counter implements the state register with a counter that can (i) count up, (ii) be loaded from a ROM, or (iii) be reset. You may assume that LOAD/CLEAR is synchronous to the clock and takes priority over the count signal.

Show the contents of your ROM and how you have wired up the COUNT, LOAD, and CLEAR signals on the diagram on the next page. Also show how the output signals A, B, and C are generated. (15 pts).



14

Cond. job 9



9. The clock cycle (i.e., the microcycle) of a CPU datapath is usually constrained by the delays incurred in one of three critical paths. Your job in this problem is to identify (for each path) the critical sequence of events that must occur within one cycle. Also, for each path, discuss one element in the path with respect to minimizing its effect on the cycle time.

36

Path 1. (HINT: Use of virtual memory is often a factor)

(5)

MAR → VA to PA translation unit → Memory → data path

VA to PA trans could cause several memory refs to access page tables.

5/

A cache of mappings (TLB) is used to reduce the need for extra memory references.

If a processor-memory cache is also used, if the page size is smaller than that (# sets) * (# bytes/line) then TLB + cache operations can't go in parallel (Phys Addr cache).

→ Make page size large enough for parallel lookup.

Path 2. (HINT: The size of the scratchpad register file is a factor)

(5)

scratchpad → bus → ALU → bus → scratchpad.

Large scratchpad has long access time.

→ Never try to do this complete cycle (scrpad → scrpad)

5/

Instead have a temp reg's at ALU output or inputs.

The scratchpad is used only to hold values needed across machine instruction boundaries. Transfers to & from scratchpad can be in parallel with other operations (eg. ALU).

Path 3. (HINT: If the machine has few instructions, this is NOT a factor)

(5)

μPC reg. → μbranch control logic → μROM → μPC reg.

The μROM can be sped-up if made smaller.

5/

This is done by having only μstate information be encoded in it, and then have the μPC reg go through a second ROM (nano ROM) to generate the Vcontrol signals.

data path

10. Draw a "register-level" diagram of a cache memory system to support a high speed CPU. Show sufficient details so that it can be seen how each feature specified below can operate. You may assume that your components are random access static memories modules, bus multiplexors, tri-state bus drivers, ALUs, registers, and ordinary logic gates and flip-flops. Show explicitly the number of bits used in each block and justify your choice. (30 pts).

FEATURE/PARAMETER SPECIFICATION:

Assume machine is word addressable

- Two-way set associative
- Write back → needs dirty bit
- FIFO replacement
- Two words/line → block size = 2 words
- 4K total words of data in the cache
- 16 bits/word
- 16 bit address bus (64K address space)

Cache needs Valid bit : (∃ valid entry?)

one per block → dirty bit (Does this entry need to be written to memory on replacement?)

one per set → FIFO bit (simple way to determine which to replace on a miss is to keep track of oldest & replace it. This is easy to do in 2-way set-asso. cache: there are only 2 possibilities)

4K total words + 2 words/line + set size = 2 ⇒ $4K/2/2 = 1K = 2^{10}$ entries

⇒ 10 bits determine set

1 bit for word in line ⇒ "word bit"?

5 bits for tag. (total = 16)

Operations Done by cache controller FSM.

Write

Word in cache → write in block, set dirty bit

not in cache →

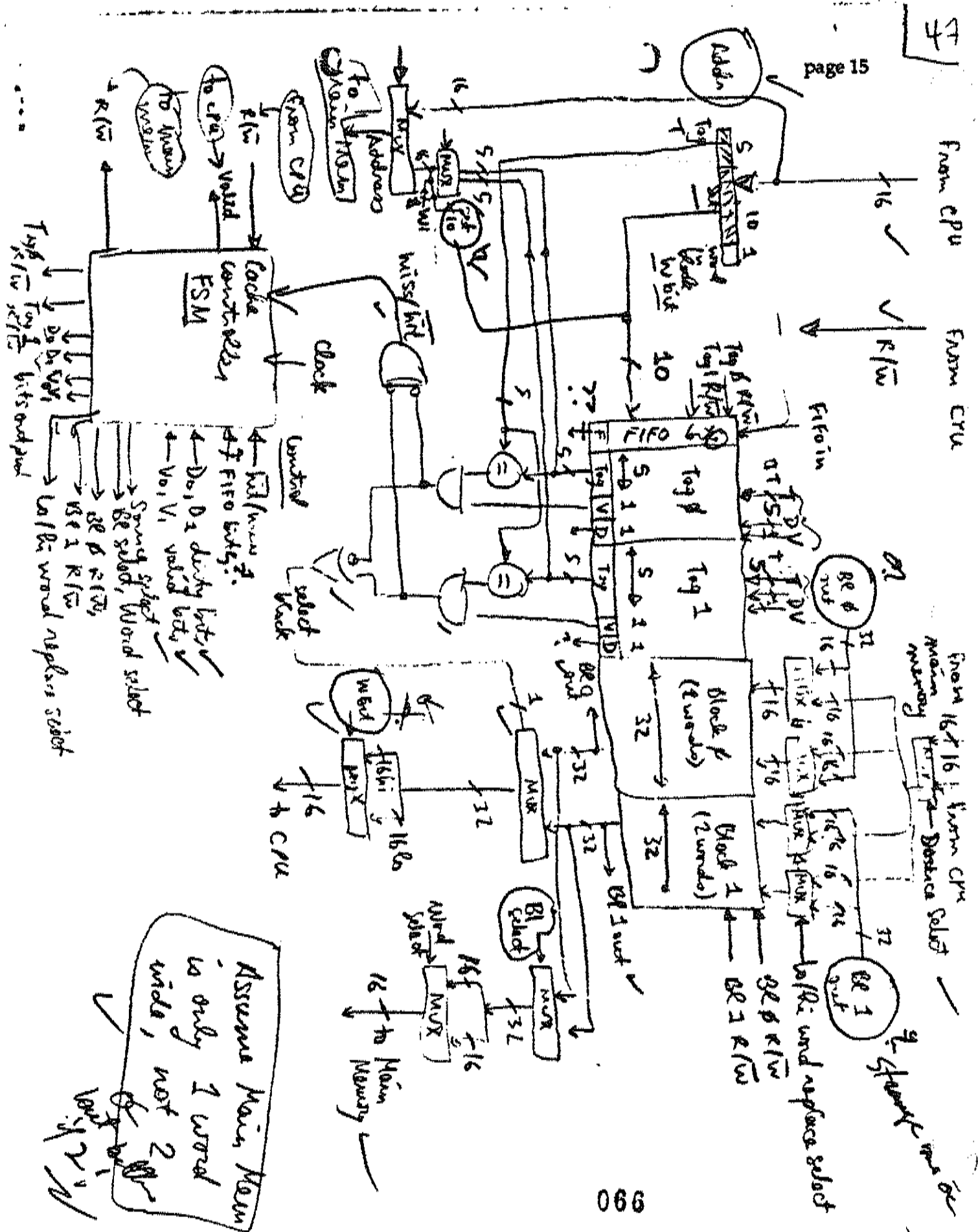
- if Dirty bit set, save block determined by FIFO bit
- write to cache, set dirty bit, set VALID bit
- change FIFO bit

Read

Word in cache → pass it on to CPU

not in cache →

- if dirty bit set, save block determined by FIFO bit
- read new store word in block determined by FIFO bit
- change FIFO bit, clear dirty bit, set VALID bit



Assume Main Mem is only 1 word wide, not 2

Hardware Core Exam: Spring 1987

067

February 9, 1987

HARDWARE PRELIMINARY EXAM

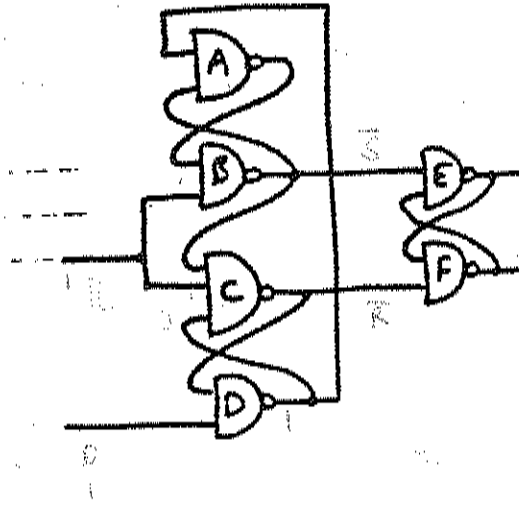
ANSWER ALL 8 QUESTIONS

ALL QUESTIONS HAVE EQUAL CREDIT

If you think any question is ambiguous, please
clearly indicate what assumption you are making
to resolve the ambiguity.

YOUR NUMBER _____

QUESTION 1:



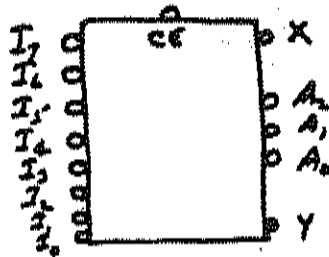
- a) What is the circuit drawn above?
- b) Name two distinguishing external features (advantages) of this circuit.
- c) What is the function of the gate labeled A ?
- d) What is the function of the gate labeled D ?
- e) What is the function of the gate labeled E ?

Your Number _____

QUESTION 2:

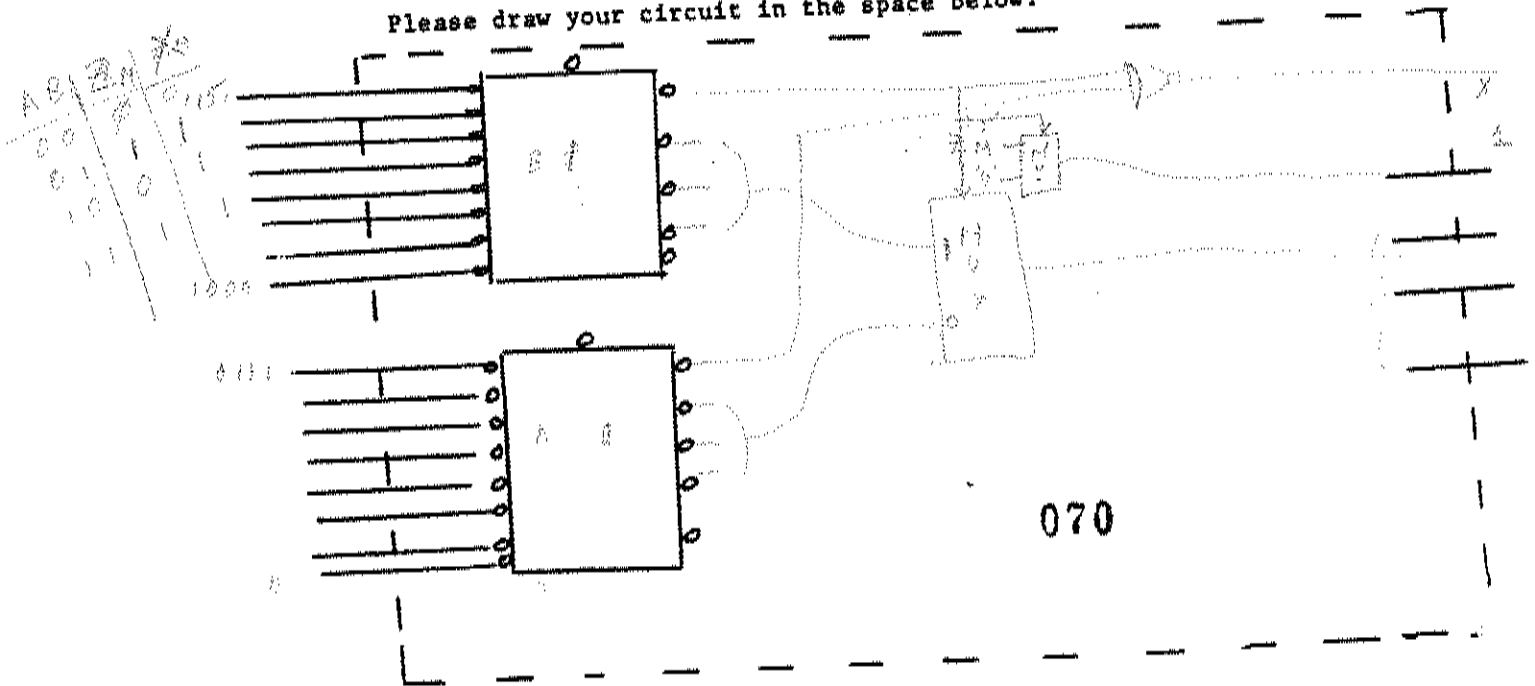
A 8-way priority encoder produces at its output the three-bit code corresponding to the highest priority input that is

asserted. The package (shown below), has eight input pins, three output pins to identify the highest priority asserted input and power, ground, and chip select pins. The package also provides two outputs X, Y.



a) What are the functions of these outputs X, Y

b) Use two of these chips plus whatever minimal additional logic gate you need to design a circuit for a 16-way priority encoder. Please draw your circuit in the space below:



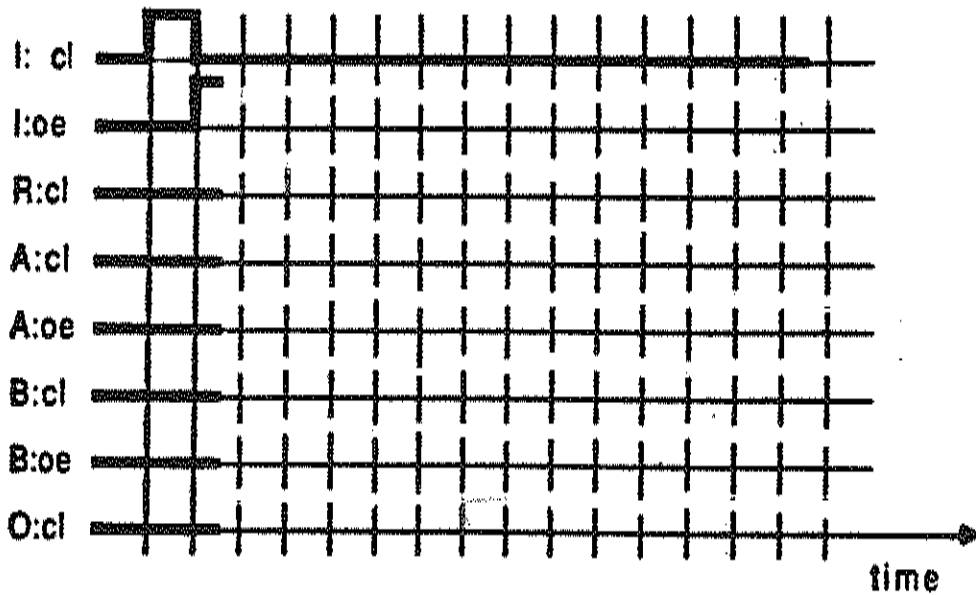
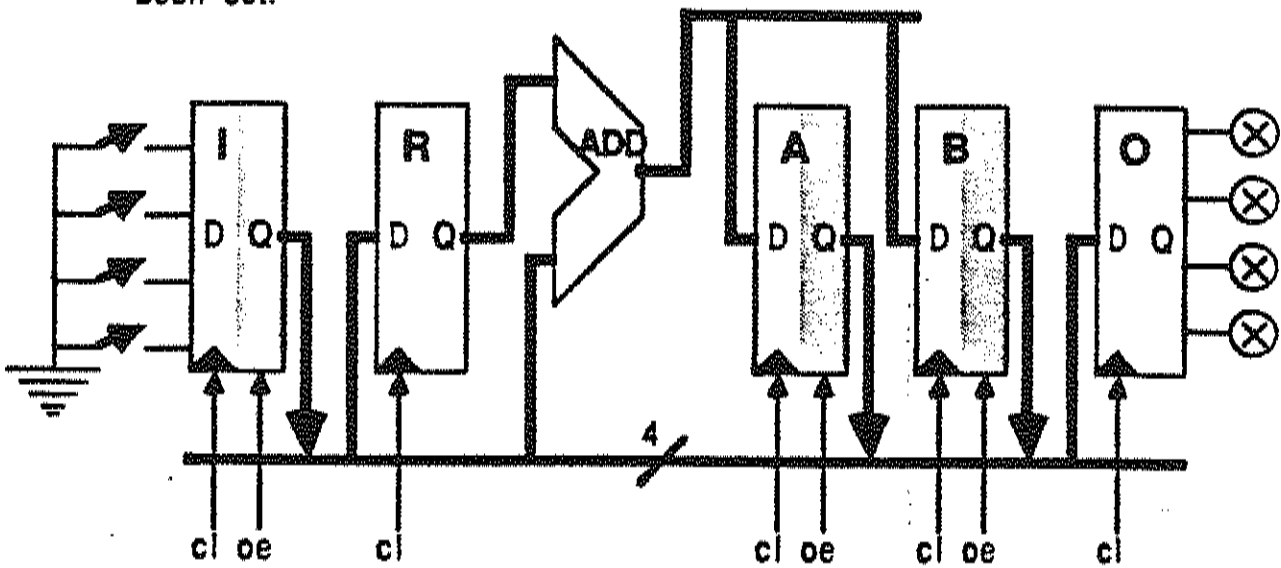
070

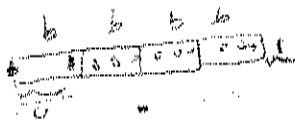
QUESTION 3:

Assume that you have a bus-connected assembly of an adder and five registers as shown below. All registers are positive edge triggered, and registers I, A, B have tri-state output. All busses are 4 bits wide.

You want to add a number *i* (which you set with the toggle switches attached to the input register I) to the number *b* (which is already contained in register B) and then display the result on the LEDs attached to the output register O.

Specify a minimal sequence of appropriate operations for the control signals *cl* (=clock input) and *oe* (=output enable) by completing the timing diagram below. Assume that the input switches have already been set.





11111111
10000000

Your Number _____

-1.0000010×2^7

2 1111100000

11111111

11111111
11111111
11111111

QUESTION 4:

Assume floating point numbers are represented in conventional normalized signed-magnitude format using n bits, where k bits are used, for the exponent. Assume the radix of the exponent is 2^b and $n-k-1$ is a multiple of b .

- a) Let y be a number which cannot be represented exactly in the above scheme, and let x be the number obtained by rounding y to the nearest representable number. What is the maximum relative error due to roundings assuming $0 < x, y$.

what happens when you round up or down?

$\cdot 000,0000$
 $\cdot 00011111$

2^b

$\cdot 00010000$
 $\cdot 00011111$
 $\cdot 00010000$
 $\cdot 00011111$
 $\cdot 00010000$
 $\cdot 00011111$
 $\cdot 00010000$
 $\cdot 00011111$

- b) Consider a particular representable number x , ($x > 0$). If y is rounded to x and $y > x$, then $x = y - e_1$. Let E_1 be the maximum value that e_1 can have. If y is rounded to x and $y < x$, then $x = y + e_2$. Let E_2 be the maximum value that e_2 can have. For most representable number x , $E_1 / E_2 = 1$

11111111
00010000
00010000
00010000
00010000

b1) When is this not the case?

0.1111111100011111

0.00010000

11111111

b2) What is the value of E_1 / E_2 for these cases?

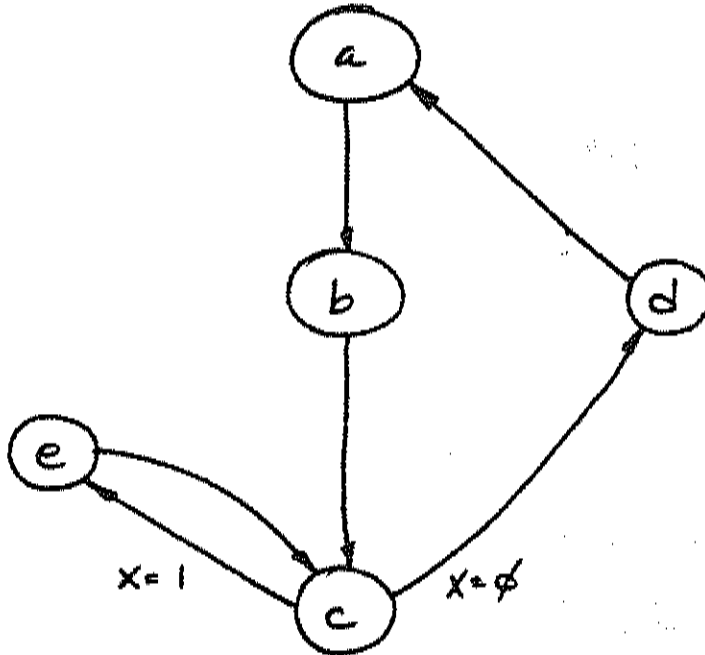
1+2-b

Your Number _____

QUESTION 5:

Implement a controller for the following state diagram using a counter with load, increment and reset lines as the main part.

Full credit will be given only to designs with minimal logic



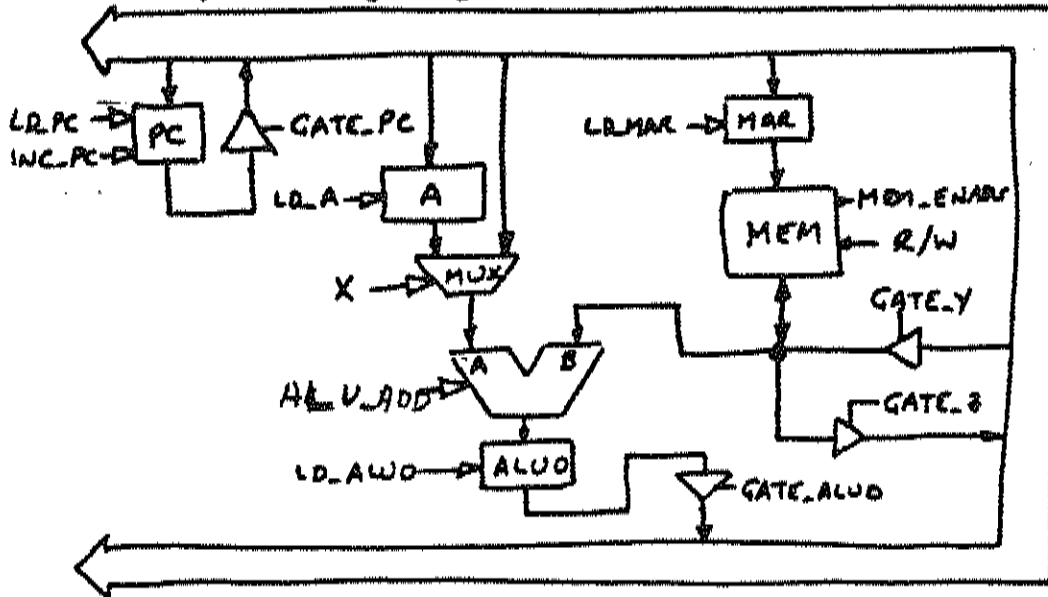
a: 0 0 0
 b: 0 0 0
 c: 1 0 0
 d: 0 0 1
 e: 0 1 0
 b: 0 1 1



QUESTION 6:

Your Number _____

The Data Path shown below is to be used as an 8-bit host machine for emulating an 8-bit processor. Assume A, MAR, and ALUO are 74 LS377 (see accompanying data sheet) which run off the same clock. Note that the LO microorders are each tied to pin G of the corresponding part. Assume memory can operate in one cycle. Assume the cycle is long enough to allow propagation delays to complete.



Your job in this assignment is to write in the space provided below the minimal microprogram to complete the emulation of the three address ADD instruction. An example is shown below, i.e. the instruction starting at location 205 causes the contents of location 64 and the contents of location 128 to be added and the sum stored in location 32. Note that the first operand is the destination address. The opcode for a three-address ADD is 00110011. Operand addresses are specified by direct addressing. Assume you are starting with PC containing 206 and the microprogram has jumped to your first microinstruction.

*gate PC,
LD PC, load MAR:
MEM_ENABLE, R, gate z
LD_A, LD_MAR
MEM_ENABLE, R, Gate Y
LD_A, INC PC
GATE PC, LD_MAR
MEM_ENABLE, R,
GATE_ALUO, LD_MAR
ALU_ADD = X + A
GATE_ALUO, GATE Y, MEM_ENABLE*

00110011	205
00100000	206
01000000	207
10000000	208

Your number _____

TYPES 8841337, 8841337A, 8841337B, 8841337C, 8841337D, 8841337E, 8841337F, 8841337G, 8841337H, 8841337I, 8841337J, 8841337K, 8841337L, 8841337M, 8841337N, 8841337P, 8841337Q, 8841337R, 8841337S, 8841337T, 8841337U, 8841337V, 8841337W, 8841337X, 8841337Y, 8841337Z
DUAL IN-LINE AND QUAD 8-TYPE FLIP-FLOPS WITH ENABLE

Standard logic diagram

Standard logic diagram showing a flip-flop circuit with inputs A, B, C, D, E, F, G, H and outputs Q1, Q2, Q3, Q4.

Standard logic diagram

Standard logic diagram

Standard logic diagram showing a flip-flop circuit with inputs A, B, C, D, E, F, G, H and outputs Q1, Q2, Q3, Q4.

Standard logic diagram

Standard logic diagram

Standard logic diagram showing a flip-flop circuit with inputs A, B, C, D, E, F, G, H and outputs Q1, Q2, Q3, Q4.

Standard logic diagram

TYPES 8841337, 8841337A, 8841337B, 8841337C, 8841337D, 8841337E, 8841337F, 8841337G, 8841337H, 8841337I, 8841337J, 8841337K, 8841337L, 8841337M, 8841337N, 8841337P, 8841337Q, 8841337R, 8841337S, 8841337T, 8841337U, 8841337V, 8841337W, 8841337X, 8841337Y, 8841337Z
DUAL IN-LINE AND QUAD 8-TYPE FLIP-FLOPS WITH ENABLE

Standard logic diagram

Standard logic diagram showing a flip-flop circuit with inputs A, B, C, D, E, F, G, H and outputs Q1, Q2, Q3, Q4.

Standard logic diagram

Standard logic diagram

Standard logic diagram showing a flip-flop circuit with inputs A, B, C, D, E, F, G, H and outputs Q1, Q2, Q3, Q4.

Standard logic diagram

Standard logic diagram

Standard logic diagram showing a flip-flop circuit with inputs A, B, C, D, E, F, G, H and outputs Q1, Q2, Q3, Q4.

Standard logic diagram

TYPES 8841337, 8841337A, 8841337B, 8841337C, 8841337D, 8841337E, 8841337F, 8841337G, 8841337H, 8841337I, 8841337J, 8841337K, 8841337L, 8841337M, 8841337N, 8841337P, 8841337Q, 8841337R, 8841337S, 8841337T, 8841337U, 8841337V, 8841337W, 8841337X, 8841337Y, 8841337Z
DUAL IN-LINE AND QUAD 8-TYPE FLIP-FLOPS WITH ENABLE

Standard logic diagram

Standard logic diagram showing a flip-flop circuit with inputs A, B, C, D, E, F, G, H and outputs Q1, Q2, Q3, Q4.

Standard logic diagram

Standard logic diagram

Standard logic diagram showing a flip-flop circuit with inputs A, B, C, D, E, F, G, H and outputs Q1, Q2, Q3, Q4.

Standard logic diagram

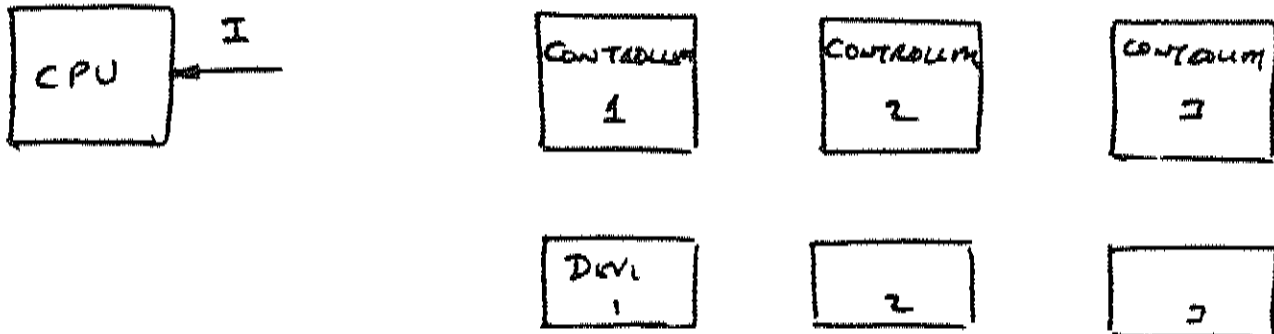
Standard logic diagram

Standard logic diagram showing a flip-flop circuit with inputs A, B, C, D, E, F, G, H and outputs Q1, Q2, Q3, Q4.

Standard logic diagram

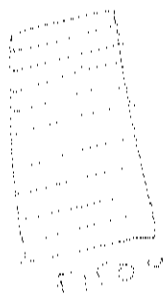
QUESTION 7:

a) Consider a collection of 3 I/O devices and their associated controllers as shown in the picture below. There is a single interrupt line, I, that is input to the CPU.



Assuming that we want the priority of device 3 higher than device 2 which is in turn higher than device 1, suggest a simple way to achieve this effect without resorting to any polling techniques.

b) Consider an I/O device which is a "buffered disk", i.e. it is a conventional magnetic disk with a controller that has X bytes of RAM local to the controller which can be used as a cache for disk blocks.



Handwritten note: Full Buffer

Your Number _____

c) What would be reasonable cache management policy and why?

Handwritten note: LRU

d) Discuss the relative merits of putting X bytes of memory in the buffered disk versus adding the same amount of memory to that available to a CPU managed buffer pool.

QUESTION 8:

Your Number _____

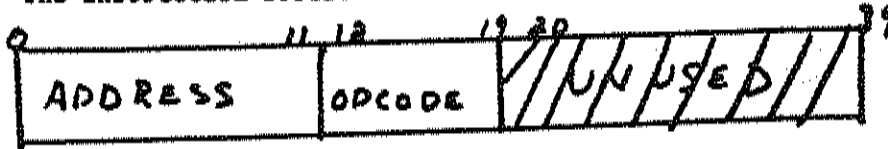
The IAS Machine created by Von Neumann had the following instruction set (This has been simplified a bit).

M [0:4095] (0:39), AC (0:39), Q (0:39)

1. LOAD	$AC \leftarrow M[x]$	x is the address field
2. LOAD NEG	$AC \leftarrow -M[x]$	
3. LOAD ABS	$AC \leftarrow M[x] $	Absolute Value
4. LOAD NEG ABS	$AC \leftarrow - M[x] $	Negative absolute value
5. ADD	$AC \leftarrow AC + M[x]$	
6. SUB	$AC \leftarrow AC - M[x]$	
7. ADD ABS	$AC \leftarrow AC + M[x] $	
8. SUB ABS	$AC \leftarrow AC - M[x] $	
9. LOAD Q	$Q \leftarrow M[x]$	
10. XFER Q	$AC \leftarrow Q$	
11. MULT	$AC:Q \leftarrow AC * M[x]$	
12. DIV	$Q \leftarrow AC/M[x]$	$AC \leftarrow \text{remainder}(AC, M[x])$
13. BRANCH	$PC \leftarrow X$	
14. BRANCH 0	If $AC \geq 0$ then $PC \leftarrow X$	
15. STORE	$M[x] \leftarrow AC$	
16. STORE IN ADDRESS	$M[x] (0:11) \leftarrow AC (0:11)$	
17. MUL 2	$AC \leftarrow AC * 2$	
18. DIV 2	$AC \leftarrow AC / 2$	

Your Number _____

The Instruction format looked like this:



- a) From this description, which number representation would you suspect is used
- (a) Sign and Magnitude
 - (b) 1's Complement
 - (c) 2's Complement

Why?

- b) As one of the first computers, it was lacking in feature.
- b1) What is the purpose of instruction 16? (Store in address)
- b2) What addition to the IAS Machine would make this instruction unnecessary?
- b3) If we ignore the problem in Question 2, what is the single most important missing instruction from the IAS Machine?

Why?

Your Number _____

b4) Define that instruction in the hardware description language
below:

Hardware Core Exam: Fall 1987

HARDWARE PRELIM EXAM, FALL 1987

- Please sign your name in the sign-up sheet before you take the exam.
- Please write your identification number on every sheet. If you attach additional sheets, please write down your I.D. number, problem number, and page number.
- Please answer all questions. Look over all the questions before starting. Questions 2, 3, 8, and 9 carry more weight than others. The approximate time to answer the questions is given.
- If you get bogged down, go to another question. Always do the ones you know how to solve first, then go back and work on the others.
- If you do not understand any part of the question, interpret it the best way you can. State your interpretation and any assumptions you may make.
- If you feel additional assumptions are necessary, keep them simple and straightforward.

GOOD LUCK!

Problem 1
15 minutes

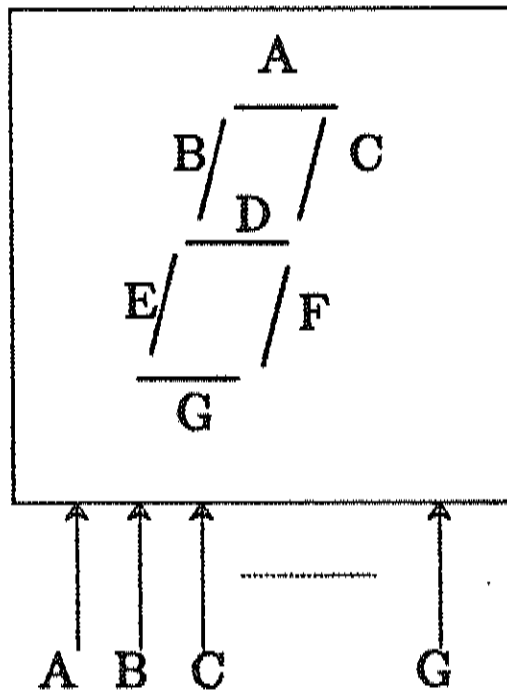
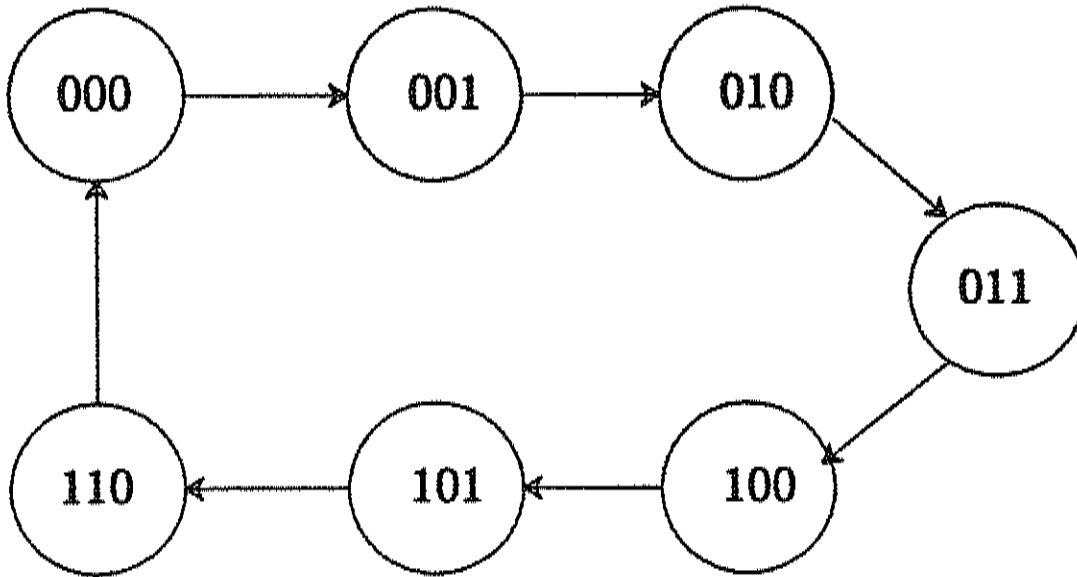
I.D. # _____

Using D FF's and assorted gates, design a sequential comparator that is to determine which of the two n -bit positive numbers, X and Y , is larger. (Assume FF's are clocked).

Problem 2
20 minutes

I.D. #

Obtain a scale-of-seven up counter, as shown in the following state diagram, using D FFs and PLA. Assume that this counter is tied to a seven segment display device. You don't have to simplify the PLA.



Design a general purpose register using 3 clocked T FF's for the following functions (using NAND gates).

Control Signal		Function
S	Set A,B,C to 1's;	111 \rightarrow ABC
D	Decrement by one binary	ABC - 1 \rightarrow ABC
L	left shift end around by one position	ABC \rightarrow BCA
W	Transfer (write) the states of input lines a,b,c, into A,B,C.	a \rightarrow A; b \rightarrow B c \rightarrow C

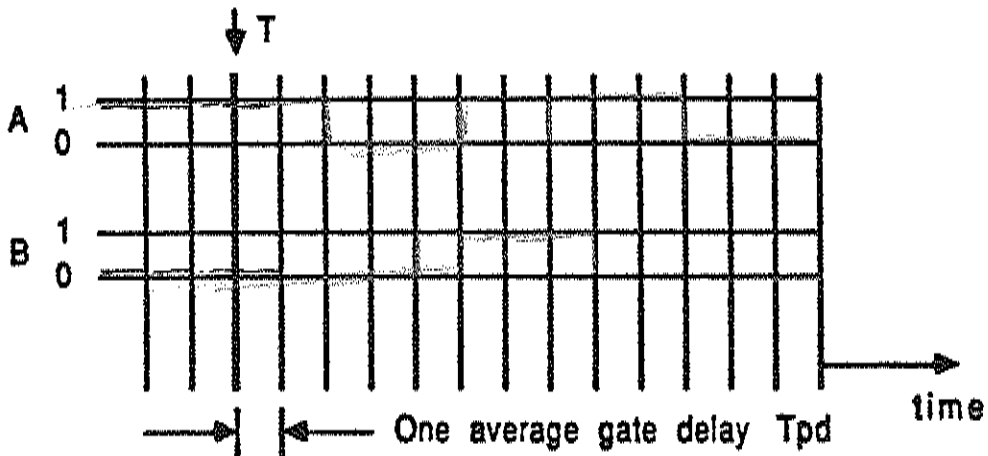
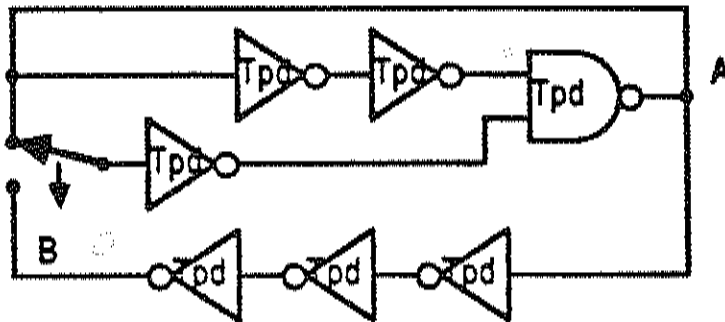
a. Derive the input (excitation) equation for each FF. b. Test your design for cases:

- i. ABC = 111
- ii. ABC = 101

Problem 4
15 minutes

I.D. #

Sketch in the timing diagram below the logic values of the signals at points A and B in the given circuit. Assume that the circuit is already in a steady state (you need to figure out what it is). At time T, the switch is moved from position UP to position DOWN.



Problem 5
10 minutes

I.D. # -----



Given a Mealy machine with 5 inputs, 6 state-flipflops, and 12 outputs.

Assuming suitable internal wiring of the machine,

a) what is the maximum possible number of different output patterns that this machine can produce ?

b) what is the maximum number of transition arrows that can leave a single state ?

c) what is the minimum number of transition arrows that can end in a single state ?

Problem 6
10 minutes

I.D. #

An Instruction Set Architecture has a 32 bit VA, and byte addressing. It is to be used with 8 MB of physical memory. We wish to design a 2-way set associative write-through virtual cache. Line size (also called block size) is 8 bytes. The Tag Store is to be indexed with 9 bits from the address.

* The Tag Store must be, therefore:

512 x bits

Specify in the space below the fields comprising one of those 512 entries. Specify the size of each field.

Assume a lifo line replacement policy. Specify below the number of bits required to accommodate that policy, what each bit pattern means, and an algorithm for what to do on Access and on Replacement.

Bit Pattern	Interpretation

On Access:	On Replacement:

Problem 7
15 minutes

I.D. #

Many machines represent integers with three different types: 2's complement, floating point, and BCD. In the space below, show the representations of the decimal number 125. (Assume 2's complement is 32 bits; floating point is 32 bits, signed-magnitude, exponent field is 8 bits, excess-128 code, radix + 2.)

	$0.11111010110100001101000011$	
--	--------------------------------	--

2's complement

floating point

BCD

Each of the three data types has its own advantages/disadvantages. In the space below, succinctly state advantages/disadvantages of each. Include an application for each.

2's complement

floating point

BCD

	2's complement	floating point	BCD
Adv.			
DisAdv.			
Appl.			

COMPARING INSTRUCTION SETS

Your task is to compare the memory efficiency of four different styles of instruction sets for two code sequences

The styles are:

- Accumulator** The source/destination operand must be the accumulator and the other operand is in memory.
- Memory-Memory** The operands are in memory, and the instructions have three operands per instruction.
- Stack** All operations occur on top of the stack; data must be placed onto the stack before an arithmetic operation. (This is the traditional stack instruction set; no fancy operations please.)
- Load-Store Reg-Reg** All operations occur in registers; data must be placed into a register before an arithmetic operation. Memory is accessed only through loads and stores, but register-to-register instructions have three operands per instruction. Assume there are 16 general purpose registers.

To measure memory efficiency, make the following assumptions:

- All opcodes of all four instruction sets are the same size: 1 byte (8 bits)
- All memory addresses (when necessary) of all four instruction sets are the same size: 2 bytes (16 bits)
- All register specifiers (when necessary) of all four instruction sets are the same size: 0.5 byte (4 bits)
- All data operands (when necessary) of all four instruction sets are the same size: 4 bytes (32 bits)

To calculate data memory traffic, assume the stack has two top-of-stack registers, but there is no other optimization that will reduce data memory traffic beyond what is already in the four instruction sets.

Inventing your own assembly language mnemonics (which is part of the question), write the equivalent assembly language code for this high-level language specification for both sequences. Assume that all the operands—A, B, C, D—are in memory. Write the best code for each case.

(a) $A := B + C;$

```

Accumulator
LOAD A, B
ADD C
STORE A, A

```

```

Memory-Memory
ADD A, B, C

```

```

Stack
PUSH B
PUSH C
ADD
POP B

```

```

Load Store Reg-Reg
LOAD R1, B
LOAD R2, C
ADD R1, R2, R1
STORE R1, A

```

Instruction bytes fetched 3
Mem Data bytes fetched 3
Which is most efficient statically (code only)?

Instruction bytes fetched 1
Mem Data bytes fetched 3

Instruction bytes fetched 4
Mem Data bytes fetched 3

Instruction bytes fetched 5
Mem Data bytes fetched 5

Which is most efficient dynamically (code+data)?

(b) $A := B + C; B := A + C; D := A - B;$

```

Accumulator
LOAD A, B
ADD C
STORE A, A
ADD C
STORE A, A
LOAD A, A
SUB B
STORE A, A

```

```

Memory-Memory
ADD A, B, C
ADD B, A, C
SUB D, A, B

```

```

Stack
PUSH A
PUSH B
ADD A
PUSH C
PUSH A
ADD B
PUSH A
PUSH B
SUB D

```

```

Load-Store Reg-Reg
LOAD R1, B
LOAD R2, C
ADD R1, R2, R1
ADD R2, R1, R2
SUB R4, R1, R2
STORE R1, A
STORE R2, B
STORE R4, D

```

Instruction bytes fetched 11
Mem Data bytes fetched 11
Which is most efficient statically (code only)?

Instruction bytes fetched 3
Mem Data bytes fetched 3

Instruction bytes fetched 11
Mem Data bytes fetched 11

Instruction bytes fetched 11
Mem Data bytes fetched 11

Which is most efficient dynamically (code+data)?

Problem 9
50 minutes

I.D. # _____

Given only two component types:

- a. 2-input NAND gates,
- b. tri-state bus drivers,

but as many as you like, show a hardware design for a 32-Bit Processor with the following instruction set:

add (x): $AC \leftarrow AC + m(x)$
stor (x): $m(x) \leftarrow AC$
branch (x): IF $AC < 0$ THEN $PC \leftarrow X$.

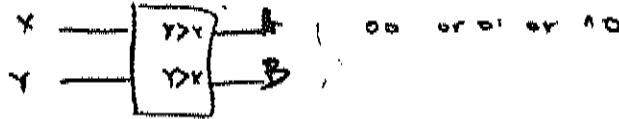
A large part of your score for this problem will depend upon your organization and neatness in presenting a solution. Use hierarchical descriptions. Use good engineering design practice in your decisions. If an assumption is needed, make the simplest assumption and state clearly what it is.

AGAIN: BE WELL ORGANIZED AND NEAT!

15/15

Using D FF's and assorted gates, design a sequential comparator that is to determine which of the two n-bit positive numbers, X and Y, is larger. (Assume FF's are clocked).

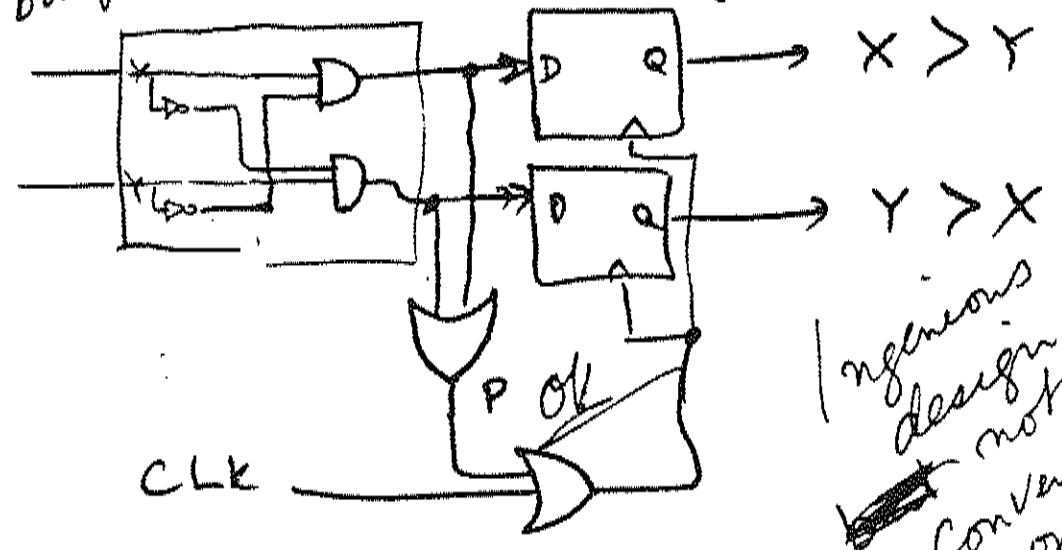
161



X	Y	X > Y	Y > X
0	0	0	0
0	1	0	1
1	0	1	0
1	1	0	0

$A = (X > Y) = X\bar{Y}$
 $B = (Y > X) = \bar{X}Y$

How far are you with start with least bit first or MSB? At any stage if $AB = 0 \Rightarrow$ continue if $AB = 01$ or $10 \Rightarrow$ stop that way not clear! From your design it is not clear



Ingenious design! not a conventional one!

if $X_i \dots X_n = Y_i \dots Y_n$ continue since $P = 0$
 then if $X_{i-1} > Y_{i-1} \Rightarrow 1st FF is set$
 (or vice versa)

093 remains set since

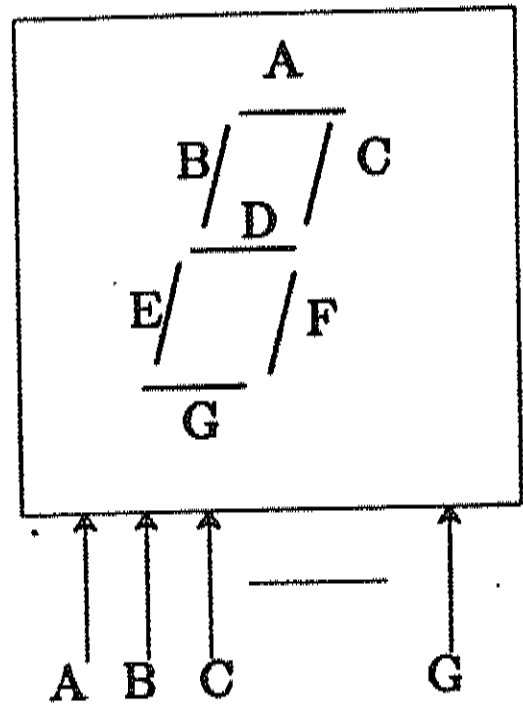
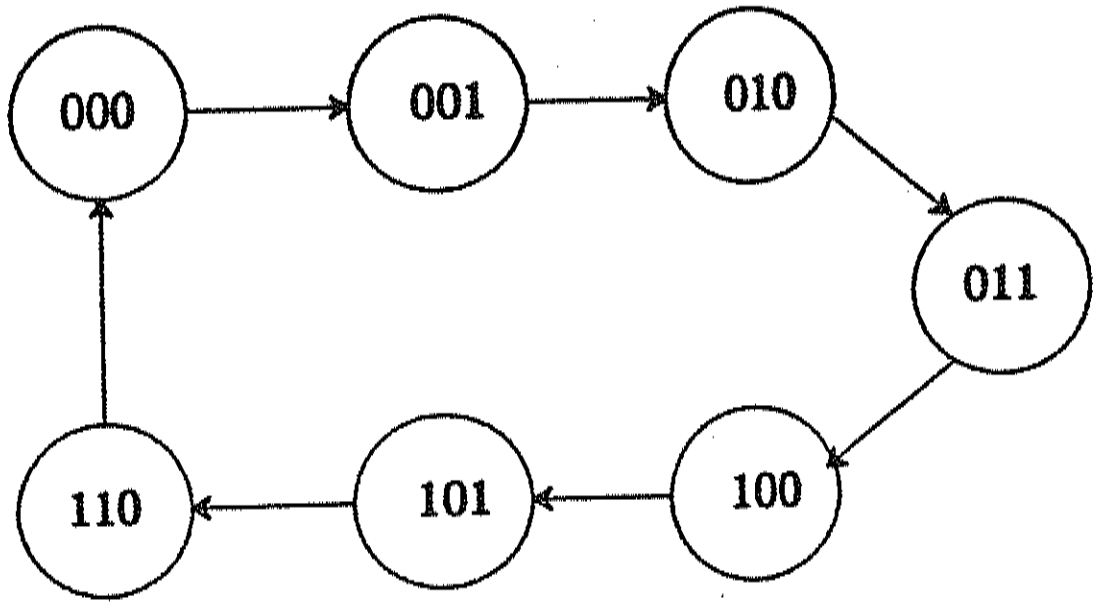
if D FF has enable input \rightarrow set \bar{P} to it. CLK = 1
Note: glitch by gating clock does not cause race problem...

18/20

ID. # 45

Problem 2
20 minutes

Obtain a scale-of-seven up counter, as shown in the following state diagram, using D FFs and PLA. Assume that this counter is tied to a seven segment display device. You don't have to simplify the PLA.



Q_2, Q_1, Q_0	Q_2^+, Q_1^+, Q_0^+
000	000
001	010
	etc...

Q_2^+

Q_2, Q_1	00	01	11	10
00	0	0	0	0
01	0	1	0	0
11	0	X	0	0
10	1	1	0	0

Q_1^+

Q_2, Q_1	00	01	11	10
00	0	1	0	0
01	X	0	X	0
11	X	X	X	0
10	0	1	0	0

Q_0^+

Q_2, Q_1	00	01	11	10
00	1	0	0	0
01	1	0	0	0
11	0	X	0	0
10	1	1	0	0

$$Q_2^+ = Q_2 \bar{Q}_1 + Q_1 Q_0 \quad \checkmark$$

$$Q_1^+ = Q_2 Q_0 + \bar{Q}_2 Q_1 \bar{Q}_0 + \bar{Q}_1 Q_0 \quad -2$$

$$Q_0^+ = \bar{Q}_2 \bar{Q}_1 + \bar{Q}_1 \bar{Q}_0 \quad \checkmark$$

Q_2, Q_1, Q_0	A	B	C	D	E	F	G
000	1	1	1	0	1	1	1
001	0	0	1	0	0	1	0
010	1	0	1	1	1	0	1
011	1	0	1	1	0	1	1
100	0	1	1	1	0	1	0
101	1	1	0	1	0	1	1
110	1	1	0	1	1	1	1

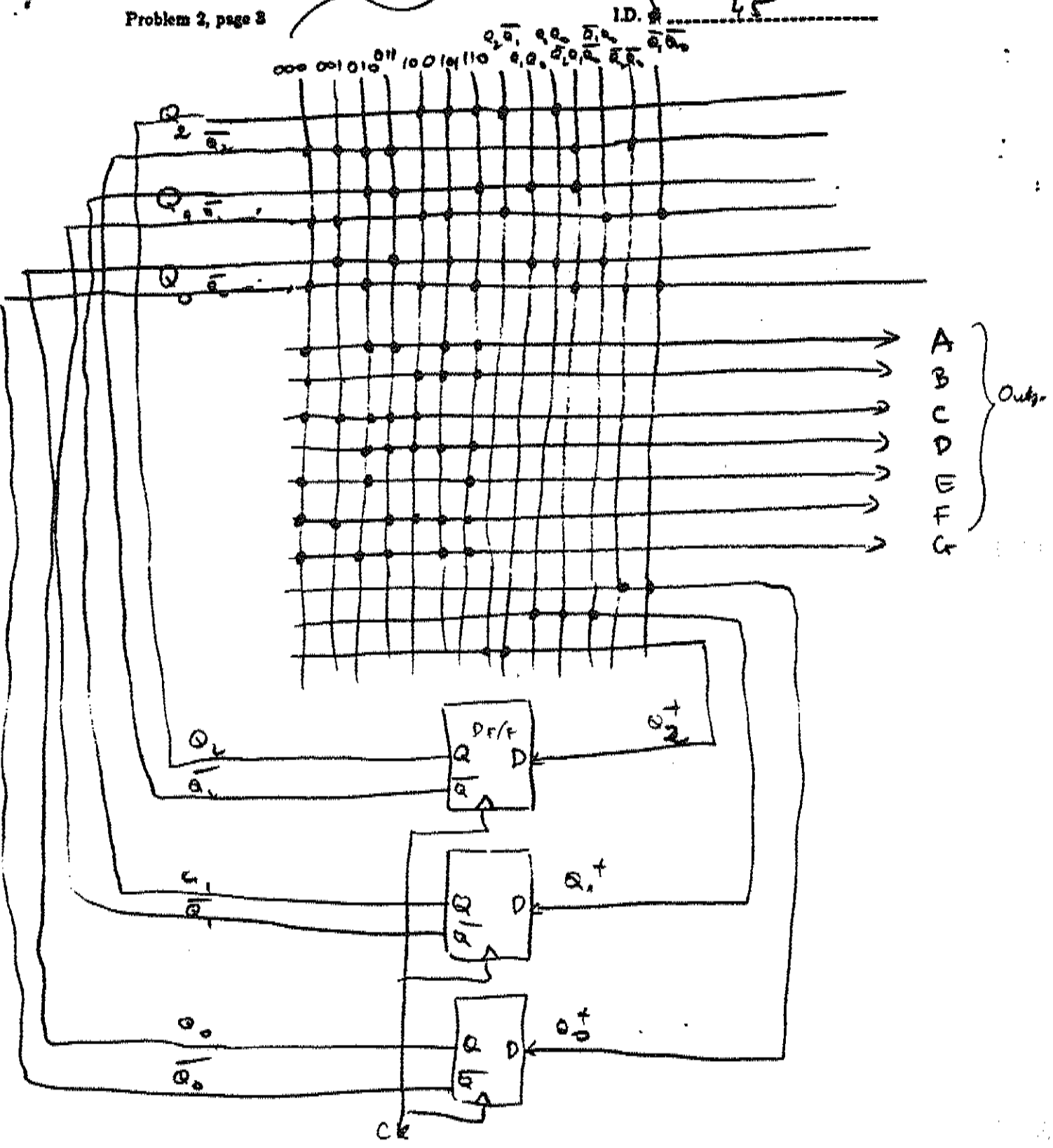
$$A = \bar{Q}_2 \bar{Q}_1 \bar{Q}_0 + \bar{Q}_2 Q_1 \bar{Q}_0 + \bar{Q}_2 Q_1 Q_0 + Q_2 \bar{Q}_1 Q_0 + Q_2 Q_1 \bar{Q}_0$$

etc... → see PLA → next page 095

minitrons

Problem 2, page 3

45



Design a general purpose register using 3 clocked T FF's for the following functions (using NAND gates).

Control Signal	Function
S	Set A,B,C to 1's; $111 \rightarrow ABC$
D	Decrement by one binary $ABC - 1 \rightarrow ABC$
L	left shift end around by one position $ABC \rightarrow BCA$
W	Transfer (write) the states of input lines a,b,c, into A,B,C. $a \rightarrow A; b \rightarrow B; c \rightarrow C$

20/20

a. Derive the input (excitation) equation for each FF. b. Test your design for cases:

- i. ABC = 111
- ii. ABC = 101

Q_2, Q_1, Q_0, D	Q_2^+	Q_1^+	Q_0^+	T_2	T_1	T_0
0 0 0	1	1	1	1	1	1
0 0 1	0	0	0	0	0	1
0 1 0	0	0	1	0	1	1
0 1 1	0	1	0	0	0	1
1 0 0	0	1	1	1	1	1
1 0 1	1	0	0	0	0	1
1 1 0	1	0	1	0	1	1
1 1 1	1	1	0	0	0	1

Q_2, Q_1, Q_0, D

0	1	0	0	0
1	1	0	0	0

$T_2 = \overline{Q_1} \overline{Q_0}$

1	0	0	1
1	0	0	1

$T_1 = \overline{Q_0}$

1	1	1	1
1	1	1	1

$T_0 = 1$

$T_2 = D \overline{Q_1} \overline{Q_0}$
 $T_1 = D \overline{Q_0}$
 $T_0 = D$

for L

0	0	0
0	1	1
1	1	1
1	0	0

0	1
0	1
0	1
0	1

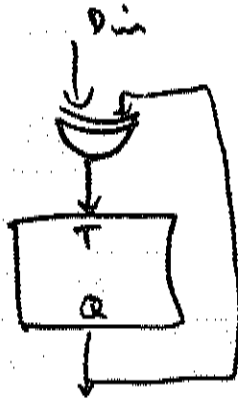
0	0
0	0
1	1
1	1

$T_2 = L Q_1; T_1 = Q_0 L; T = Q_2 L$

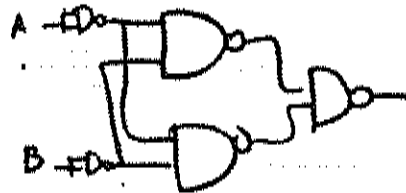
Problem 3, page 2

ID. # 45

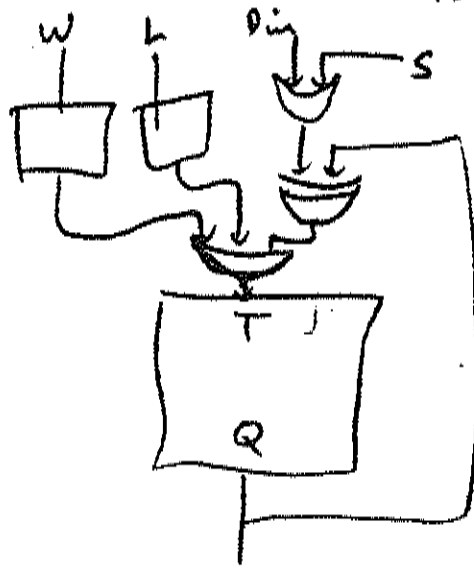
loop



where $A \oplus B = C$



so,



$$T_2 = \overline{D} \overline{a_1} \overline{a_2} + L Q_1 + (a + S) \oplus Q_2 \quad \checkmark$$

$$T_1 = \overline{D} \overline{a_0} + a_0 L + (b + S) \oplus Q_1$$

$$T_0 = D + a_2 L + (c + S) \oplus Q_0$$

^
NAND-NAND

where $\begin{matrix} A \\ B \\ C \end{matrix} \oplus \begin{matrix} P \\ Q \\ R \end{matrix} = \begin{matrix} A \\ B \\ C \end{matrix} \oplus \begin{matrix} P \\ Q \\ R \end{matrix}$

→ next page

c) $ABC = 111$

$S = 1 \Rightarrow ABC = 111$

$P = 1 \Rightarrow 110$

$L = 1 \Rightarrow 111$

$W = 1 \rightarrow abc$

cc) $ABC = 101$



$S = 1 \Rightarrow ABC = 111$

$P = 1 \rightarrow 100$

$L = 1 \Rightarrow 011$

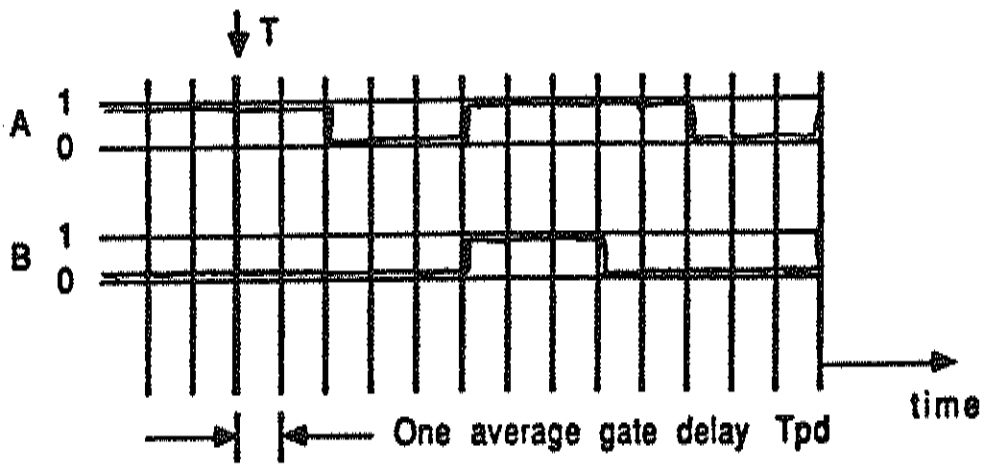
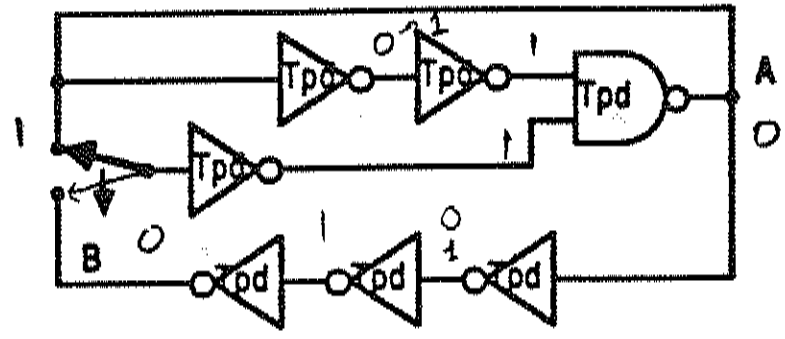
$W = 1 \rightarrow abc$

Problem 4
15 minutes

15
15

ID. # 45

Sketch in the timing diagram below the logic values of the signals at points A and B in the given circuit. Assume that the circuit is already in a steady state (you need to figure out what it is). At time T, the switch is moved from position UP to position DOWN.



Problem 6
10 minutes

10
10

ID. # 45



Given a Mealy machine with 5 inputs, 6 state-flipflops, and 12 outputs.

Assuming suitable internal wiring of the machine,

a) what is the maximum possible number of different output patterns that this machine can produce ?

$$\underline{2^5 \cdot 2^6 = 2^{11}} \quad \checkmark$$

b) what is the maximum number of transition arrows that can leave a single state ?

$$\underline{2^5 = 32} \quad \checkmark$$

c) what is the minimum number of transition arrows that can end in a single state ?

$$\underline{0} \quad \checkmark$$

Problem 6
10 minutes

ID. # 27

$\frac{10}{10}$

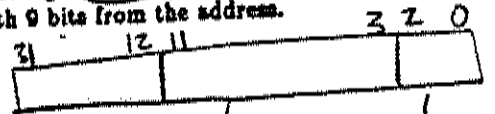
16 byte
line

An Instruction Set Architecture has a 32 bit VA, and byte addressing. It is to be used with 8 MB of physical memory. We wish to design a 2-way set associative write-through virtual cache. Line size (also called block size) is 8 bytes. The Tag Store is to be indexed with 9 bits from the address.

* The Tag Store must be, therefore:

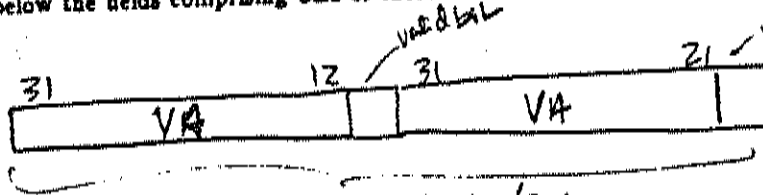
512 x 42 bits

+ 8 byte line



Specify in the space below the fields comprising one of those 512 entries. Specify the size of each field.

tag bits
for 1 set



2 entries/set

(no dirty bit — write through)

perhaps a usage bit for replacement

Assume a sfo line replacement policy. Specify below the number of bits required to accommodate that policy, what each bit pattern means, and an algorithm for what to do on Access and on Replacement.

Bit Pattern	Interpretation
have one bit that is set if the entry on the left (higher order bits) was the first in	otherwise if one on right was first in, FIFO = 0

On Access:	On Replacement:
Access is not affected as all by the FIFO bit	if FIFO = 1 then replace data on the left w/ incoming data FIFO ← 0 if FIFO = 0 then replace

I assume you are only asking about how Access was not affected by the FIFO bit, not accessing in general

(not is the FIFO bit affected by access)

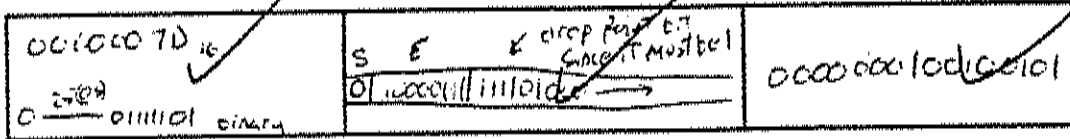
data on right w/ incoming data
FIFO ← 1

Problem 7
15 minutes

ID. # 39

14
15

Many machines represent integers with three different types: 2's complement, floating point, and BCD. In the space below, show the representations of the decimal number 125. (Assume 2's complement is 32 bits; floating point is 32 bits, signed-magnitude, exponent field is 8 bits, excess-128 code, radix + 2.)



2's complement

floating point

BCD

$0111101 = 2^7$

Each of the three data types has its own advantages/disadvantages. In the space below, succinctly state advantages/disadvantages of each. Include an application for each.

	2's complement	floating point	BCD
Adv.	Easy arithmetic signed +, - for numbers easy calculate not a wrap (as in binary)	Represents large numbers (up to almost 2^{128}) and small ones, down to 2^{-128} Like scientific notation Basic math	Maps directly into decimal for straightforward conversion has built-in extra space for the temp. of flow
DisAdv.	Size limit of numbers to -2^k up to $2^k - 1$ only integers or fixed pt	Complex notation mechanisms. Usually for approx. ints. Not exact representations of most #'s	Very space wasteful (16 bits for 1000 instead of 65535) Requires special arithmetic processing
Appl.	general signed integers for general computing (i.e. systems software) OS, etc.	Any code needing lg/sml real FP i.e. scientific appl. astronomy, etc	Business work where "exact" things are needed decimal is needed (Bank software)

Since this is a PhD exam for a core area, the grading is based on the seriousness of the implications of the mistakes rather than just getting points for filling in the blanks with the correct numbers and no points for incorrect numbers.

Data traffic was considered correct if you consistently included the writes in your count or if you consistently only counted the reads. Optimized code sequences were allowed ("D:=C") but not necessary, unless the lack of "optimization" showed a misunderstanding of the architecture.

Worst mistakes (for they simplified the question)

Lose 10 points:

- 1. Not understanding what data memory traffic is. A few broke the size into bytes of opcode and bytes of addressing. This mistake also simplified the question.
- 2. Counted instructions rather than bytes of instructions (for this mistake significantly simplified the question).

Most serious mistakes (for they suggest major misunderstandings about hardware or instruction sets)

Lose 8 points:

- 3. Not realizing you have to use a STORE to get data from the CPU to memory in a single accumulator machine or register-register machine or a POP in a stack machine.
- 4. Not realizing that an instruction set can have variable length instructions (e.g., thinking that in a accumulator instruction set both "LOAD A" and "NEG" take 3 bytes).

Next most serious mistakes:

Lose 6 points:

- 5. Not realizing that the accumulator is an implied operand (e.g., thinking that you must say "ADD A,Acc").
- 6. Not knowing that data transfer on a stack architecture (PUSH/POP) changes the stack contents.
- 7. (Apparently) Not knowing assembly language.
- 8. (Apparently) Not knowing the difference between a bit and a byte.

Serious mistakes:

Lose 4 points:

- 9. Rounding up instructions to a multiple of 8 bits (computers have been built with instruction sizes that can be an odd number of bits)
- 10. Using LOAD and STORE rather than PUSH and POP with a stack architecture.
- 11. Always using just 2 addresses in the Load-Store Reg-Reg architecture even though the question says that this is a three address machine.
- 12. Forgetting that the two top-of-stack registers in a stack architecture prevent extra memory accesses.
- 13. Ending up with a error in code size or data memory traffic count for which I had no possible explanation.

Medium mistakes: Lose 3 points:

- 14. Not leaving the stack empty after a sequence of instructions (e.g. not ending with a POP).
- 15. Forgetting that data that is still in the accumulator after a STORE and does not have to be refetched with a LOAD.
- 16. Getting fancy with duplicates in the stack case and resulting in extra data memory traffic because it exceeds the 2 TOS registers.
- 17. Misunderstanding (miscounting?) the number of data words transferred for a code sequence in an architecture.

Small mistakes: Lose 2 points:

- 18. Calculated "C:=B-A" rather than "C:=A-B". (OK if you mentioned you had a reverse subtract and used some other symbol than "SUB".)
- 19. Consistently writing the number of data words rather than data bytes (since this may affect your conclusion on which was most efficient).
- 20. Sometimes using just 2 addresses in the Load-Store Reg-Reg architecture even though the question says that this is a three address machine.

Minor mistakes: Lose 1 point:

- 21. Consistently assuming data was 2 bytes rather than 4 bytes (as stated in the question).
- 22. When they are easy for me to determine from the information on the exam, simple mistakes in counting the number of bytes in a code sequence were considered minor.
- 23. Forgetting an obvious instruction in a code sequence (e.g., a "STORE R3,A" when had "STORE R2,B").

Problem 8
25 minutes

ID. # 45

COMPARING INSTRUCTION SETS

Your task is to compare the memory efficiency of four different styles of instruction sets for two code sequences.

The styles are:

- Accumulator** The source/destination operand must be the accumulator and the other operand is in memory.
- Memory-Memory Stack** The operands are in memory, and the instructions have three operands per instruction. All operations occur on top of the stack; data must be placed onto the stack before an arithmetic operation. (This is the traditional stack instruction set; no fancy operations please.)
- Load-Store Reg-Reg** All operations occur in registers; data must be placed into a register before an arithmetic operation. Memory is accessed only through loads and stores, but register-to-register instructions have three operands per instruction. Assume there are 16 general purpose registers.

To measure memory efficiency, make the following assumptions:

- All opcodes of all four instruction sets are the same size: 1 byte (8 bits)
- All memory addresses (when necessary) of all four instruction sets are the same size: 2 bytes (16 bits)
- All register specifiers (when necessary) of all four instruction sets are the same size: 0.5 byte (4 bits)
- All data operands (when necessary) of all four instruction sets are the same size: 4 bytes (32 bits)

To calculate data memory traffic, assume the stack has two top-of-stack registers, but there is no other optimization that will reduce data memory traffic beyond what is already in the four instruction sets.

Inventing your own assembly language mnemonics (which is part of the question), write the equivalent assembly language code for this high-level language specification for both sequences. Assume that all the operands—A, B, C, D—are in memory. Write the best code for each case.

(a) $A := B + C;$

```

Accumulator
LOAD B (ALIAS)
ADD C (ACCUM-C)
STORE A (A-ACC)
    
```

```

Memory-Memory
ADD B, C, A
    
```

```

Stack
PUSH B (74-B)
PUSH C (75)
ADD (74+75)
POP A
    
```

```

Load-Store Reg-Reg
LOAD B, R1 (45) → R1
LOAD C, R2
ADD R1, R2, R3 (R1+R2 → R3)
STORE R3, A (R3 → Mem)
    
```

note add these 4 bytes of operand - write is included

Instruction bytes fetched 9 / Mem Data bytes fetched 8 / Which is most efficient statically (code only)? Mem-Mem
 Instruction bytes fetched 7 / Mem Data bytes fetched 8
 Instruction bytes fetched 10 / Mem Data bytes fetched 8
 Instruction bytes fetched 13 / Mem Data bytes fetched 8
 Which is most efficient dynamically (code+data)? Mem-Mem

(b) $A := B + C; B := A + C; D := A - B;$

```

Accumulator
LOAD B
ADD C
STORE A
ADD C
STORE B
LOAD A
SUB B
STORE D
    
```

```

Memory-Memory
ADD B, C, A
ADD A, C, B
SUB A, B, D
    
```

```

Stack
PUSH D
PUSH C
ADD
POP A → A away
PUSH A
PUSH C
ADD
POP B
PUSH A
PUSH B
SUB
POP D
    
```

```

Load-Store Reg-Reg
LOAD B, R1
LOAD C, R2
ADD R1, R2, R3
STORE R3, A
ADD R2, R3, R4
STORE R4, B
SUB R3, R4, R5
STORE R5, D
    
```

OK
 $R4 = B + A + C$
 $R5 = A - B$

Instruction bytes fetched 24 / Mem Data bytes fetched 30 / Which is most efficient statically (code only)? Mem-Mem
 Instruction bytes fetched 21 / Mem Data bytes fetched 24
 Instruction bytes fetched 30 / Mem Data bytes fetched 24
 Instruction bytes fetched 25 / Mem Data bytes fetched 8
 Which is most efficient dynamically (code+data)? Load-Store Reg-Reg

Problem 9
50 minutes

LD. #

45

44

Given only two component types:

- a. 2-input NAND gates,
- b. tri-state bus drivers,

but as many as you like, show a hardware design for a 32-Bit Processor with the following instruction set:

add (x): $AC \leftarrow AC + m(x)$
 stor (x): $m(x) \leftarrow AC$
 branch (x): IF $AC < 0$ THEN $PC \leftarrow X$.

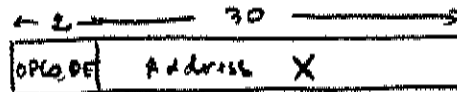
A large part of your score for this problem will depend upon your organization and neatness in presenting a solution. Use hierarchical descriptions. Use good engineering design practice in your decisions. If an assumption is needed, make the simplest assumption and state clearly what it is.

AGAIN: BE WELL ORGANIZED AND NEAT!

OpCode Table:

ADD \rightarrow 00
 STOR \rightarrow 01
 BRANCH \rightarrow 10

Instruction Format:



Assumption: Mem. can be fetched in 1 cycle

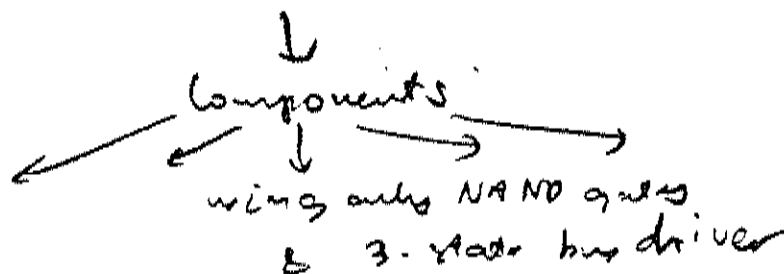
Or next page is the high level diagram of E-Unit & Control Unit

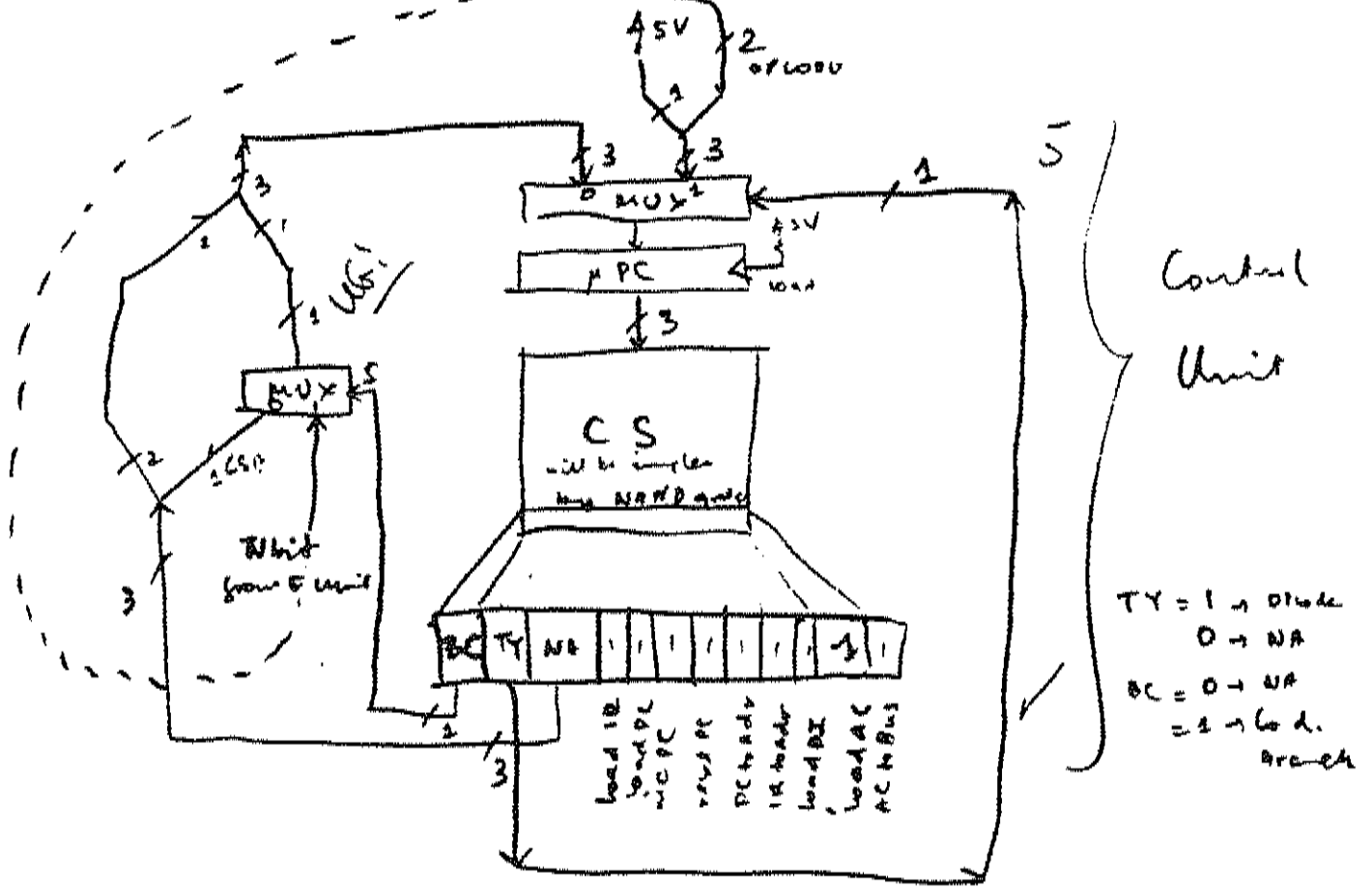
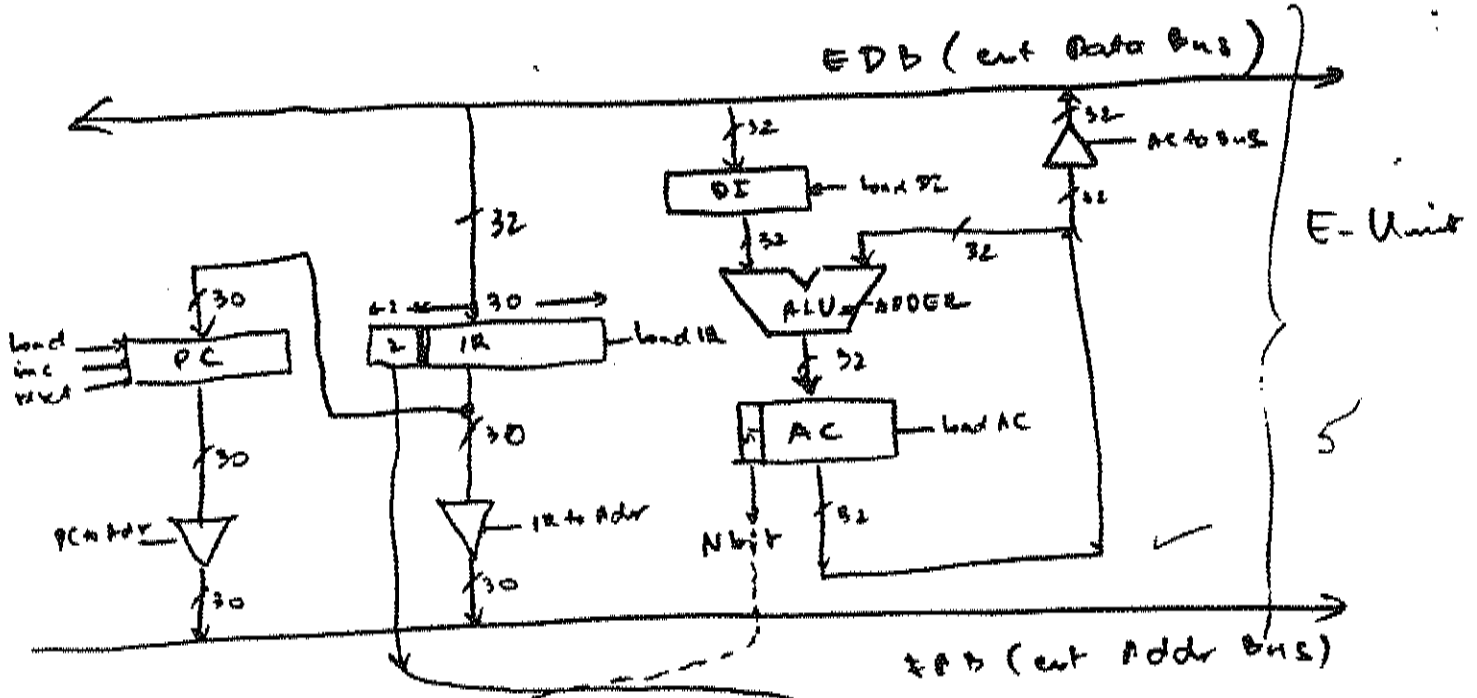
The following page is Flowchart (Diagram) ^{State}

Components design is on next 2 pages etc...

Top Down Design:

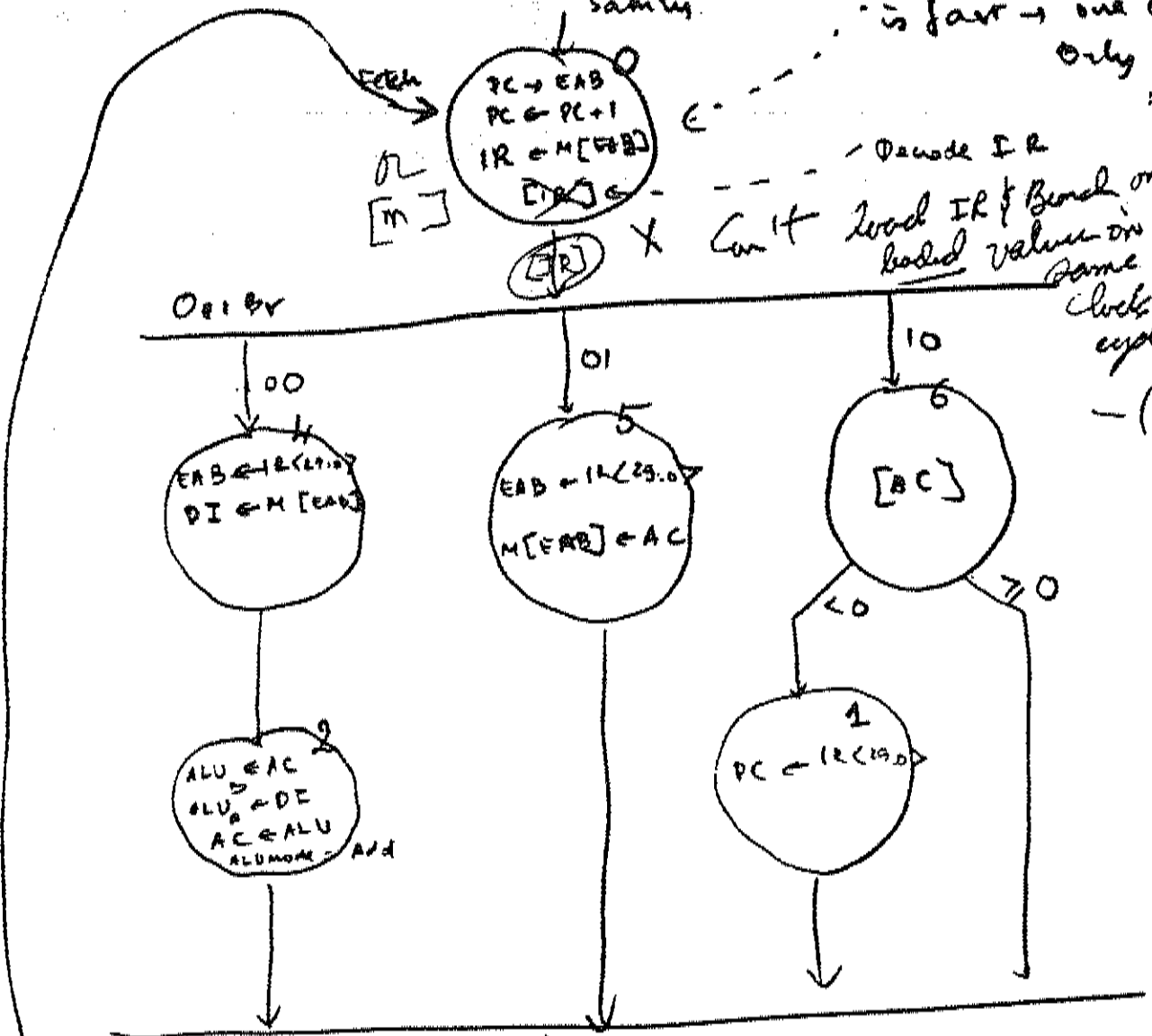
Reg Diagram + State Diagram





Flowchart
sanity.

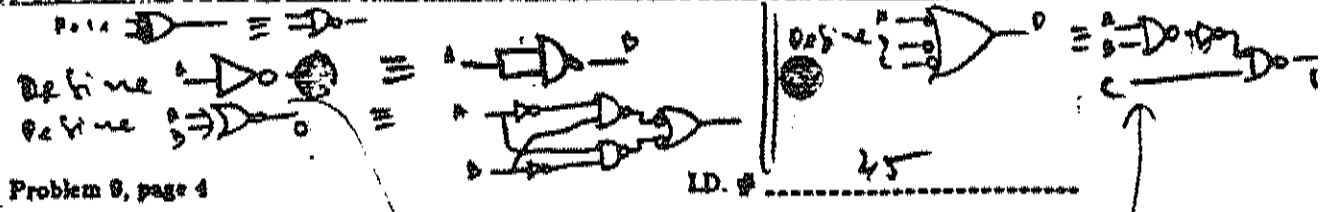
Assume that Mem
is fast → one cycle
only:



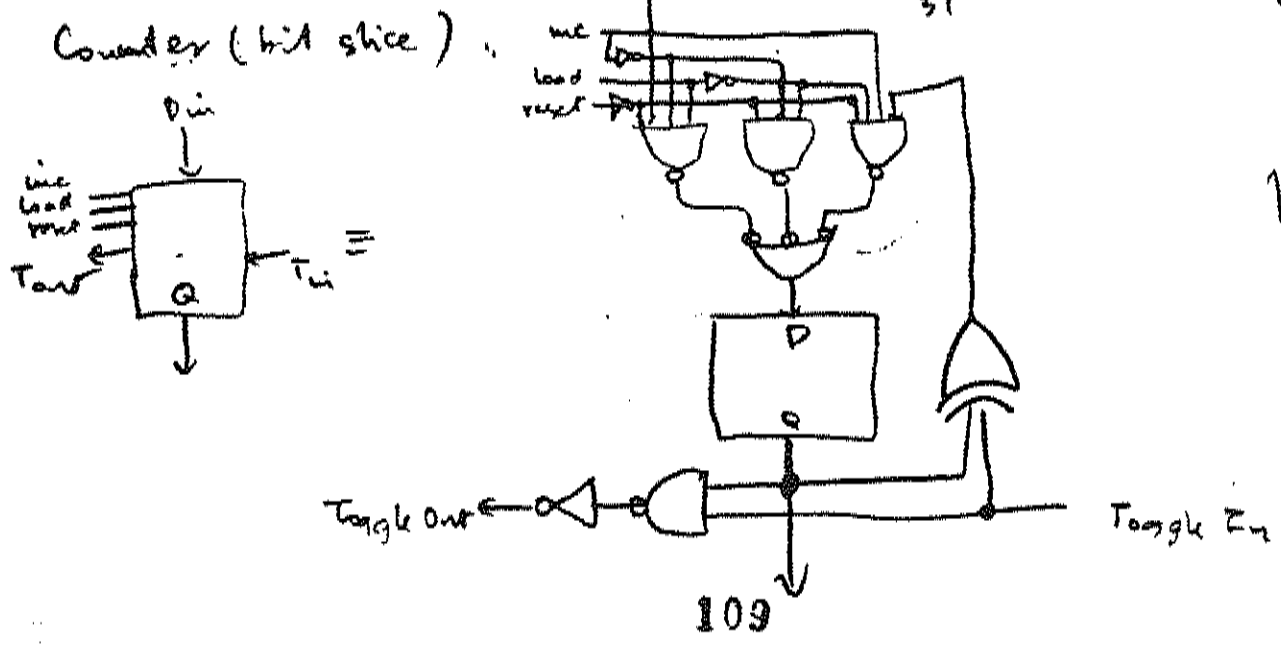
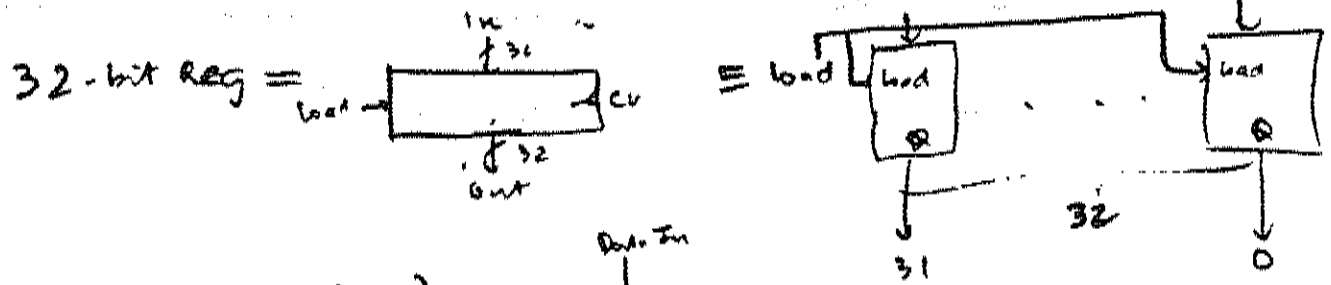
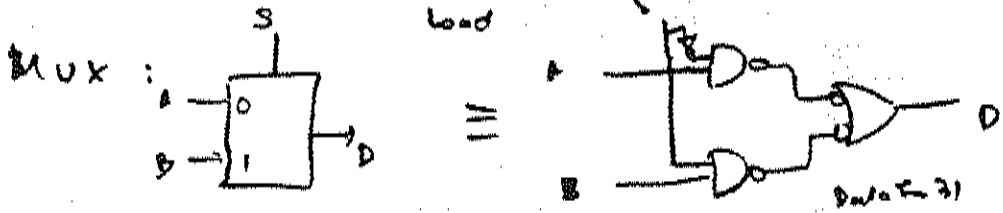
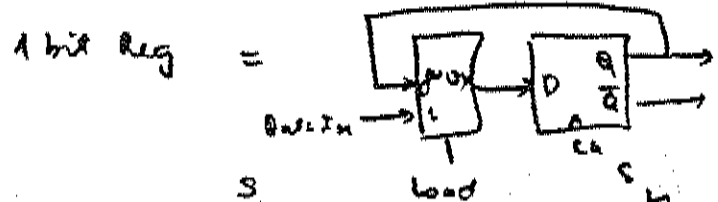
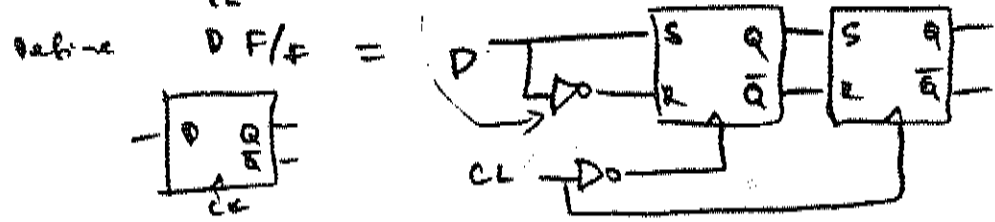
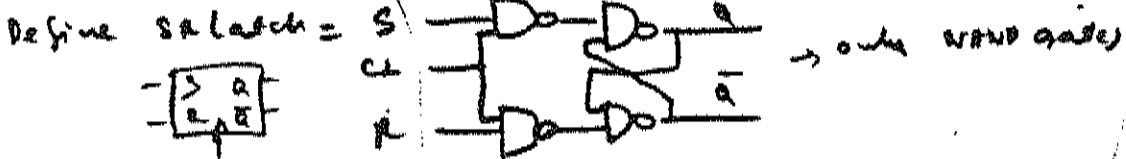
Decode IR
Load IR & Branch on
boded value on
same
clock
cycle

state ASS. → see blue pen
→

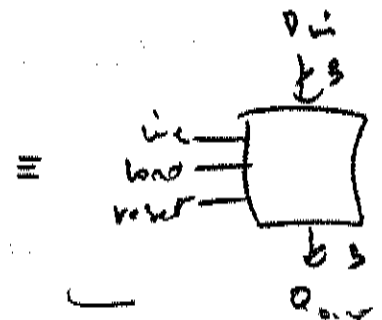
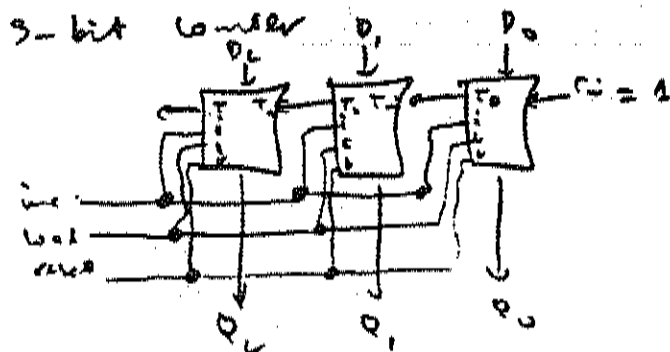
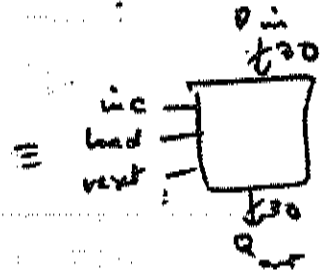
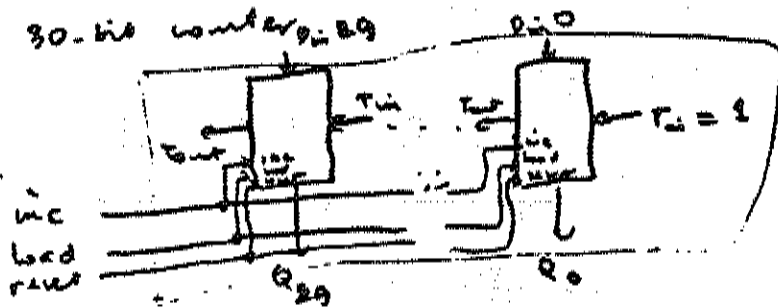
4



Problem 8, page 4



10

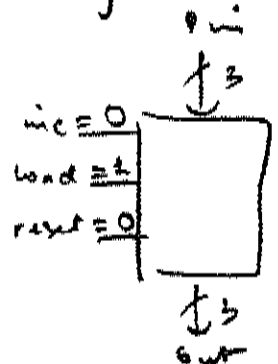


IR = DI = AC \equiv 32-bit Reg

PC = 30-bit Counter

Muxes \rightarrow already defined

μ PC = 3 bit counter with



Only things left are ALU & CS

ALU bit slice \equiv



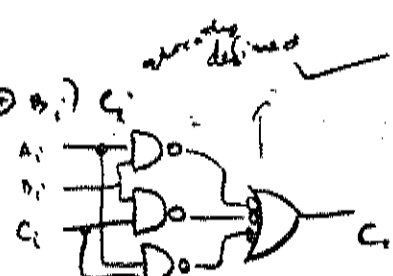
	0	1
00	0	0
01	0	1
11	1	1
10	0	1

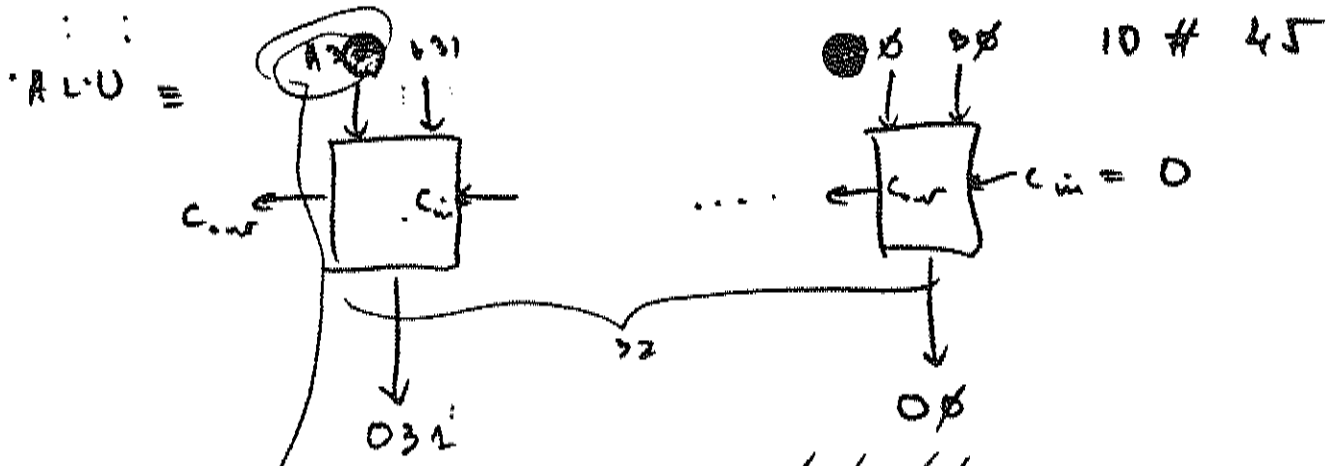
0	1
0	0
0	1
1	0

C_i

$$S_i = (A_i \oplus B_i) \oplus C_i$$

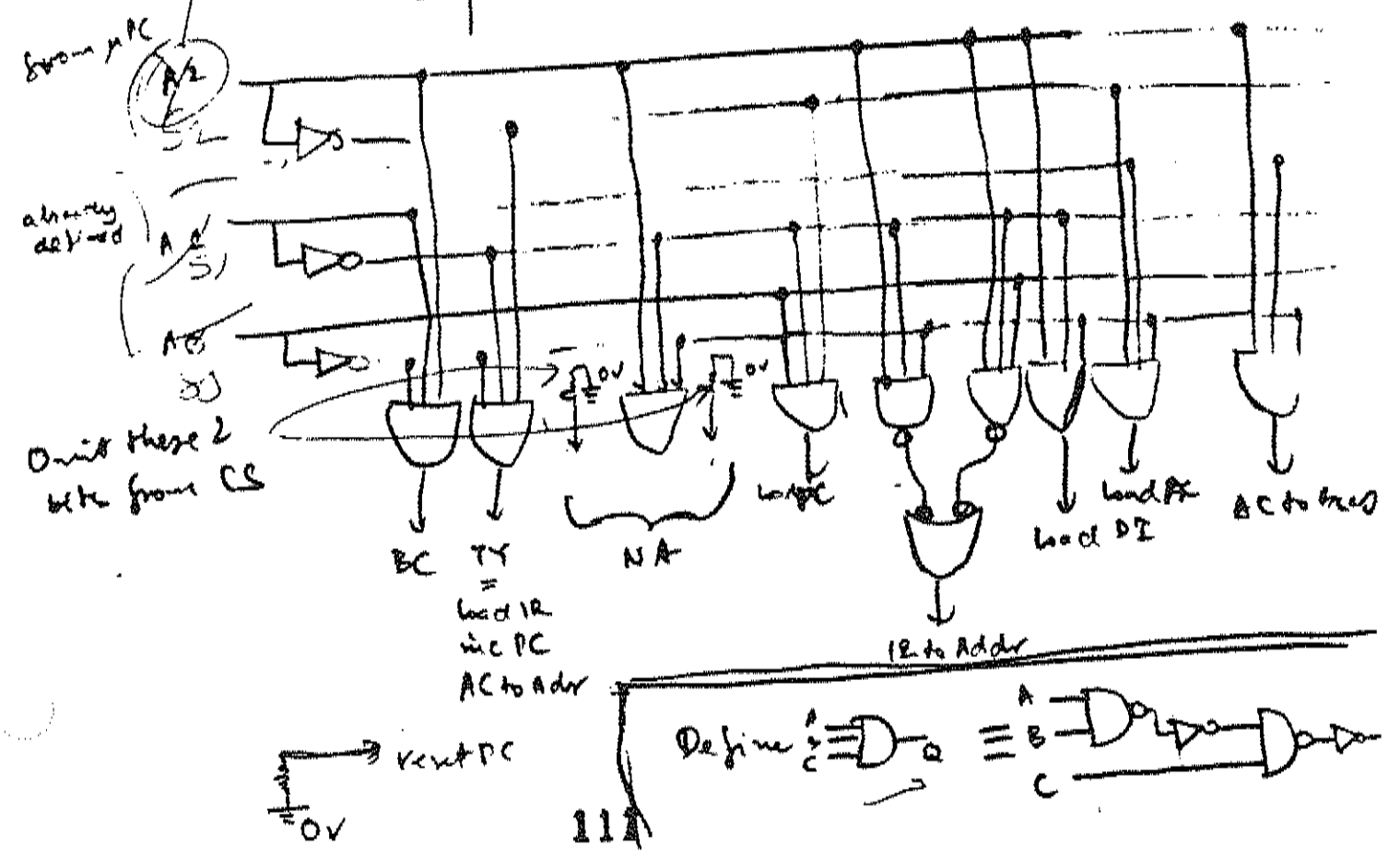
$$C_{i+1} = A_i B_i + B_i C_i + A_i C_i$$





CS:

A2A1A0	DC	TY	NA	Load IR	Load PC	PC	PC to Addr	IR to Addr	Load DI	Load AC	AC to bus
000	X	1	xyy	1	0	1	0	0	0	0	0
001	0	0	0000	1	0	0	0	0	0	0	0
010	0	0	0000	0	0	0	0	0	0	1	0
011	X										X
100	0	0	0100	0	0	0	0	1	1	0	0
101	0	0	0000	0	0	0	0	1	0	0	1
110	1	0	0000	0	0	0	0	0	0	0	0



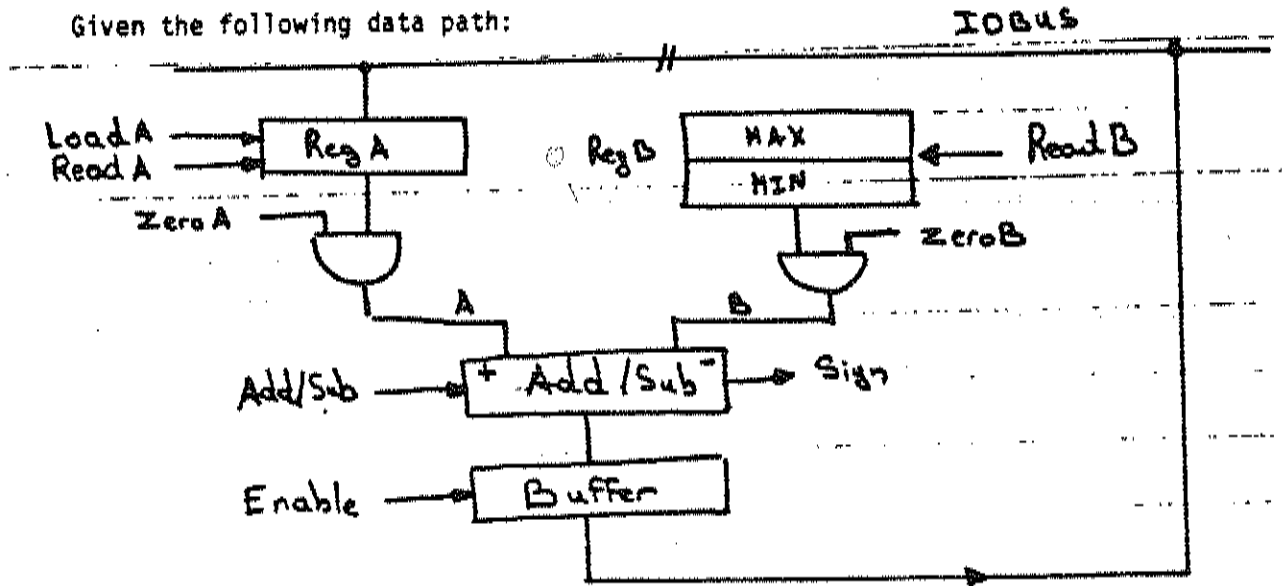
Hardware Core Exam: Spring 1988

HARDWARE PRELIM EXAM, SPRING 1988

- * Please sign your name in the sign-up sheet before you take the exam.
- * Please write your identification number on every sheet. If you attach additional sheets, please write down your I.D. number, problem number and page number.
- * Please answer all questions. Look over all the questions before starting. (The approximate time to answer the questions is given.) Questions 1 thru 6 count the same value, with question 7 counting as two questions.
- * If you get bogged down, go to another question. Always do the ones you know how to solve first, then go back and work on the others.
- * If you do not understand any part of the question, interpret it the best way you can. State your interpretation and any assumptions you may make with your answer.
- * If you feel additional assumptions are necessary, keep them simple and straightforward.

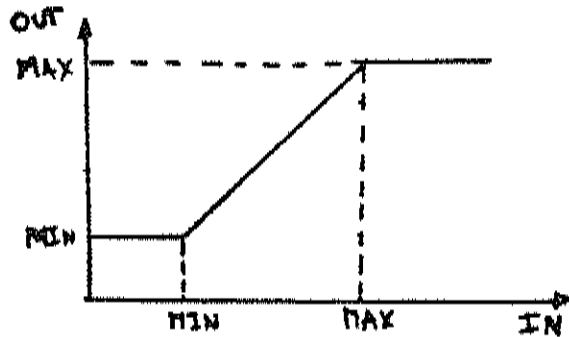
G - O - O - D L - U - C - K !!!!!

Given the following data path:



MAX and MIN are constants stored in Reg B (positions 0 and 1).

Design a controller (with complete definition of eventual rom or pla contents), which executes following function on the above data path. The function has to be performed continuously: a new execution starts at the completion of the previous one:



The status signal Sign (sign bit of the 2' complement output of the adder/subtractor) is available as an input to the controller.

Following control signals are available to determine the operation of the data path (active high):

Load A: Load value of IOBUS in Reg A

Read A: Read value of Reg A

Zero A, Zero B: Connect Reg (A,B) to adder of 1, else inject 0.

Read B: Read value of MAX of 1, else

Read value of MIN

ADD/SUB: Add (A+B) of high, else subtract (A-B)

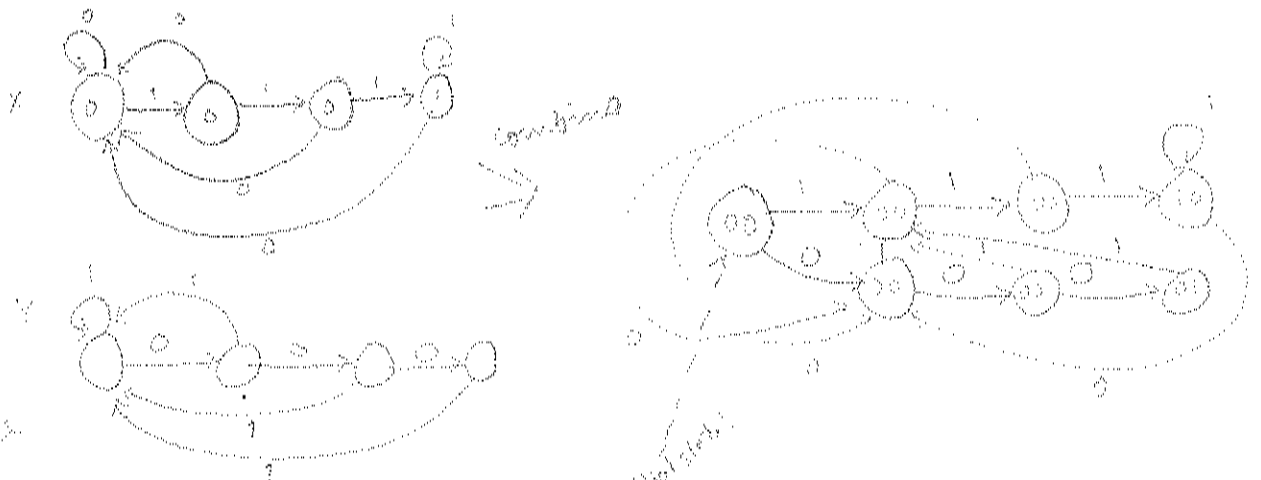
Enable: Enable Buffer Output, else tristate

High-level Design of a State Machine

Construct a synchronous Moore machine with two inputs A and B, and with two outputs X and Y. The machine accepts synchronously with the clock data on the two input lines. The output X is at a logical 1 when the data on the two inputs has been the same (A=B) for three or more consecutive clock cycles; otherwise it is at logical 0. Output Y is at logical 1 when the data on the two inputs has been exact complements for three or more cycles.

A) Give a block diagram of an implementation of such machine using MSI components: Registers, Counters, Flip-flops, Decoders, Multiplexers, Adders, Comparators, etc. Choose a straight-forward approach; no need to go into any details.

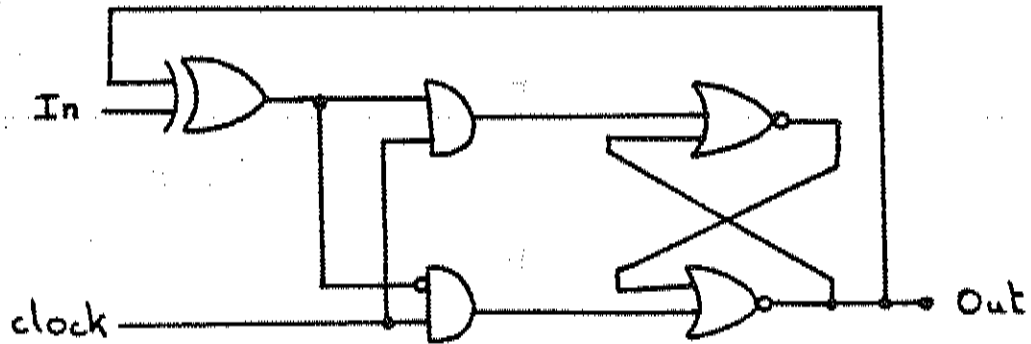
B) Draw a minimum state diagram as a formal problem description for the above machine. Clearly indicate the significance of each state and where the outputs come from.



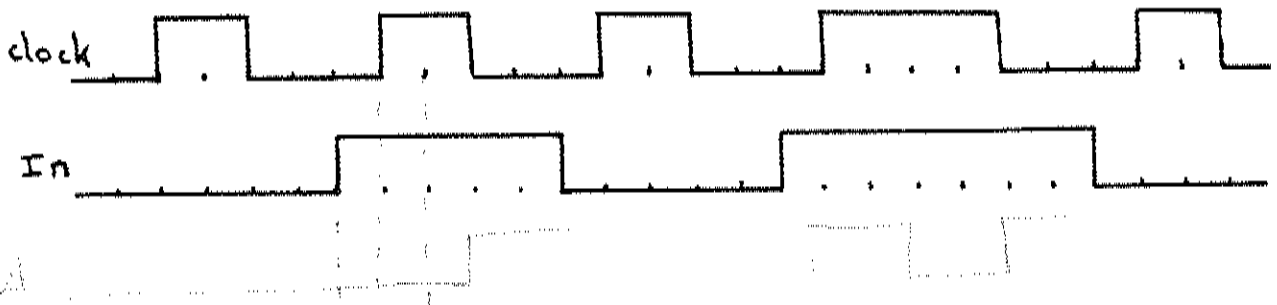
You are to design an asynchronous building block component for a hardware stack. The reusable component implements a single element of the stack. On PUSH, it reads data from the cell on its right and passes its current data to the cell on its left. POP moves data in the opposite direction. By asynchronous, we mean that each element may have its own internal clock, but that the system as a whole has no global clock. Further, PUSH and POP signals cannot be distributed globally.

- 1) Draw a black box diagram of the basic element, indicating all data and control signals it needs.
- 2) Sketch the state diagram from implementing the PUSH function.
- 3) Show a timing diagram for typical PUSH operations.
- 4) Describe how you would deal with an empty stack (underflow) or a full stack (overflow).

Given following circuit:



- A. What is the function of the above circuit?
- B. Sketch the output waveform for following clock and In signals:
Suppose that the delay of all logic elements is identical and equal to one time unit. Assume Out is initially low.

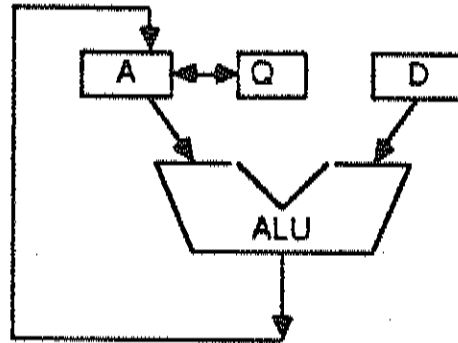


- C. Explain the abnormal behavior of the circuit.
How can this be avoided?
Redesign the circuit so that these effects disappear.

INTEGER ARITHMETIC

- A) What is the key difference between restoring and non-restoring division?

Given this portion of a simple data path and assuming non-negative operands:



- B) Is Restoring or Non-Restoring faster?
- C) Give an equation showing how much faster it is.
State your assumptions.

Problem 6
10 Minutes

ID# _____

MICROPROGRAMMING

- A) What is the key reason for using a two level control store?
- B) Suppose you are building a microprogrammed computer and the only memory chips available are 4096 word by 1 bit chips. A trial one level design requires 4096 by 100 bits (100 chips). Given this situation, in what circumstances (if any) would a two-level control store design be superior?

Given only two component types:

- A. 2-input NOR gates,
- B. tri-state bus drivers,

but as many as you like, show a hardware design for a 32-Bit Processor with following instruction set:

sub (x): $AC \leftarrow AC - m(x)$

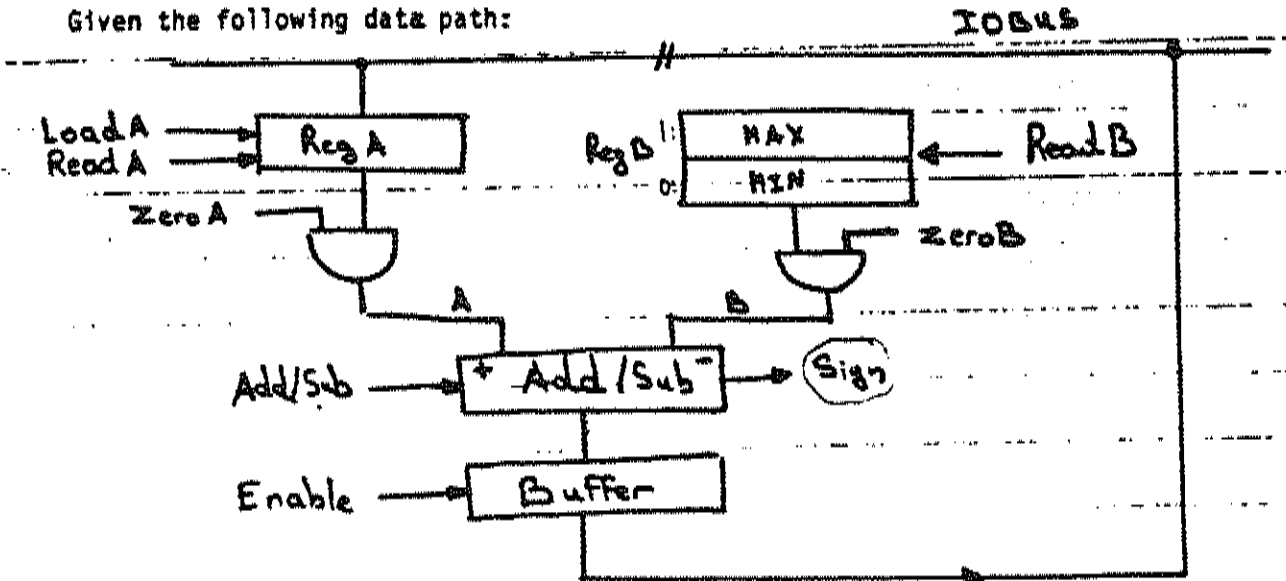
stor (x) $m(x) \leftarrow AC$

branch (x): IF $AC \geq 0$ THEN $PC \leftarrow X$.

A large part of your score for this problem will depend upon your organization and neatness in presenting a solution. Use hierarchical descriptions. Use good engineering design practice in your decisions. If an assumption is needed, make the simplest assumption and state clearly what it is.

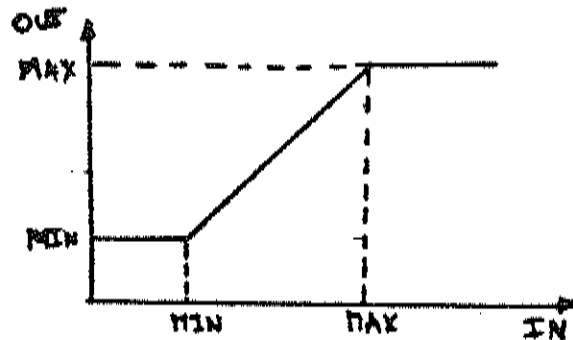
AGAIN: BE WELL ORGANIZED AND NEAT!

Given the following data path:



MAX and MIN are constants stored in Reg B (positions 0 and 1).

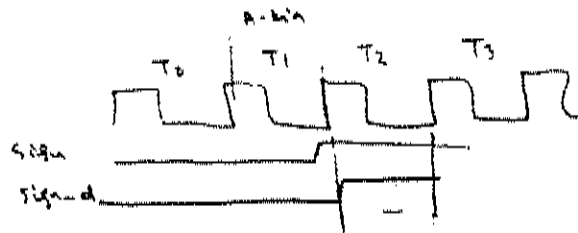
Design a controller (with complete definition of eventual rom or pla contents), which executes following function on the above data path. The function has to be performed continuously: a new execution starts at the completion of the previous one:



The status signal Sign (sign bit of the 2's complement output of the adder/subtractor) is available as an input to the controller.

Problem 1 (continued)
30 Minutes

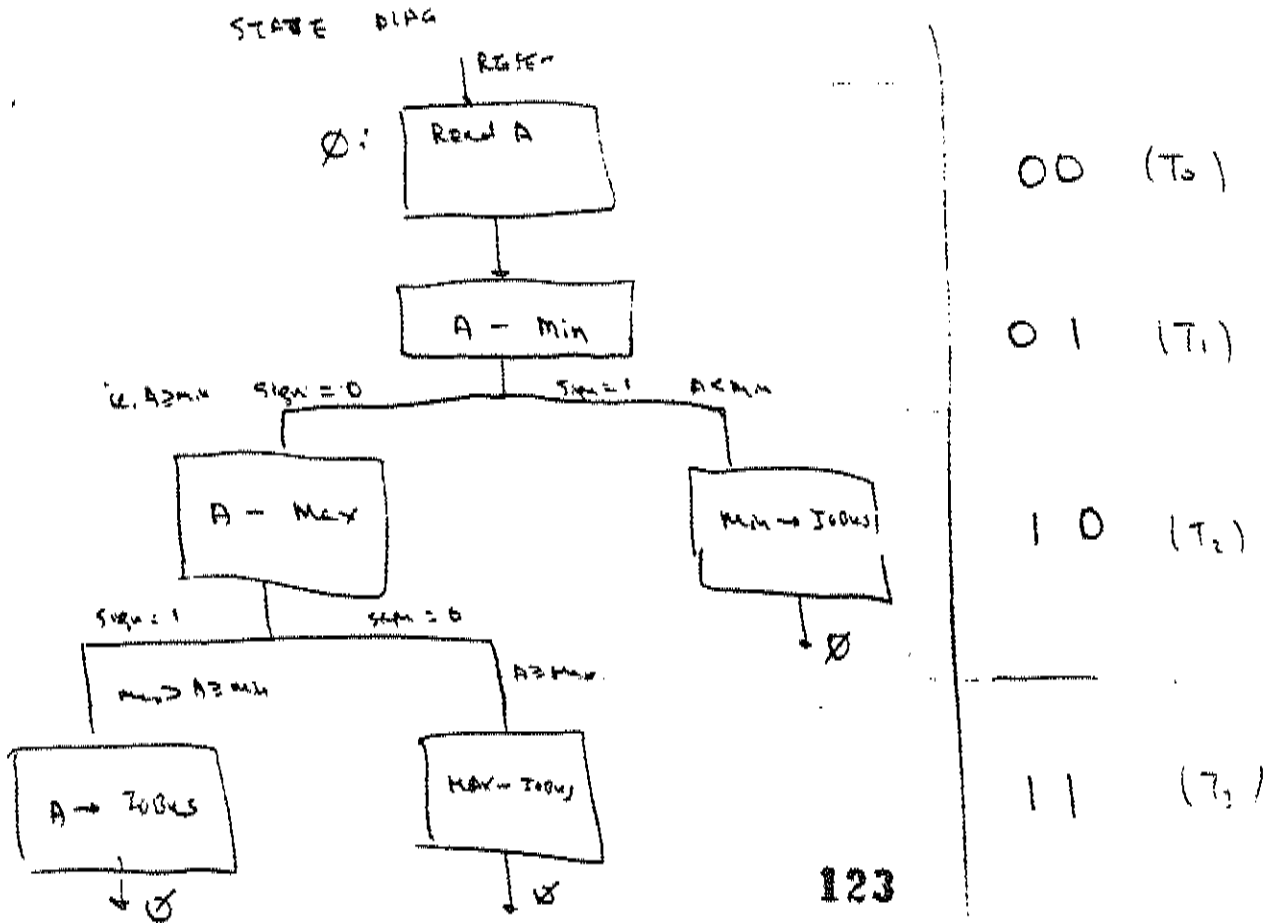
ID# 66



Following control signals are available to determine the operation of the data path (active high):

- Load A: Load value of IOBUS in Reg A
- Read A: Read value of Reg A
- Zero A, Zero B: Connect Reg (A,B) to adder of 1, else inject 0.
- Read B: Read value of MAX of 1, else :
- Read value of MIN : 0
- ADD/SUB: Add (A+B) if high, else subtract (A-B)
- Enable: Enable Buffer Output, else tristate

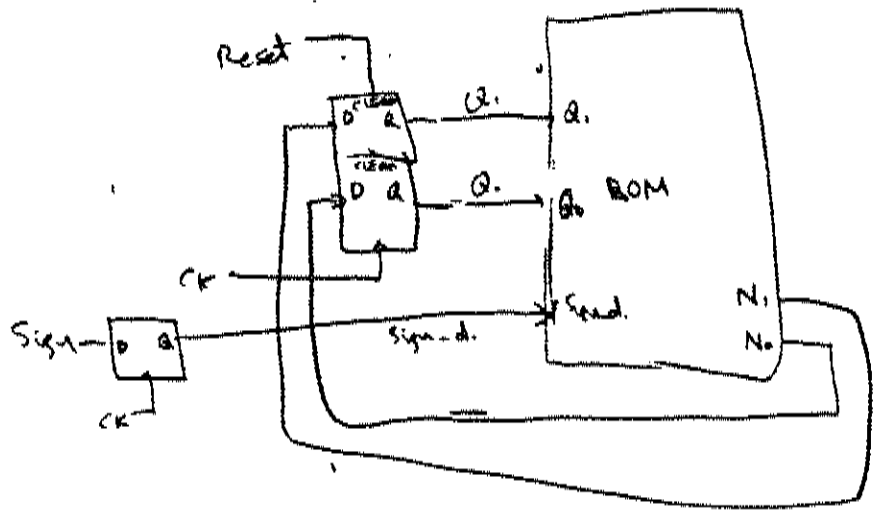
Assume IOBUS is driven with new input only when Load A = 1 by external memory



Problem 1
30 Minutes

Timing Control

A - min



Rom Table

Next state

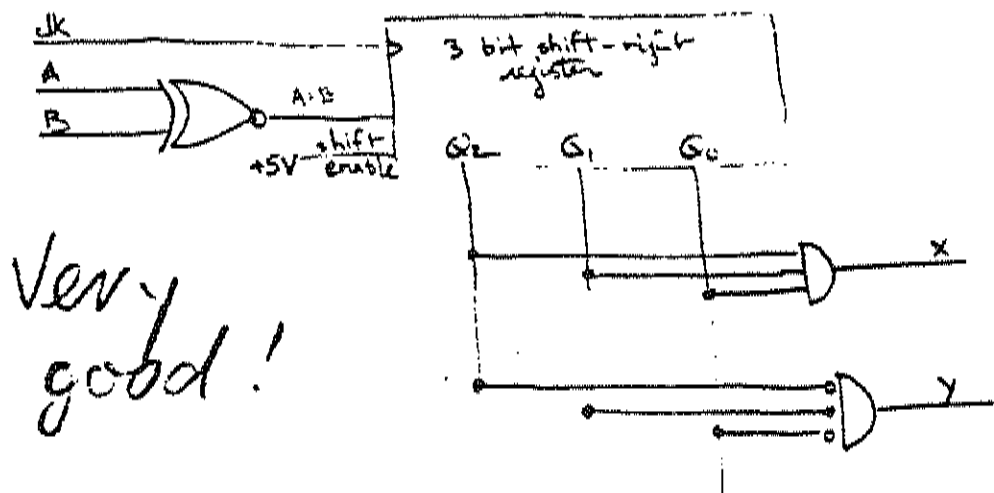
	Q1	Q0	Sign-d	N1	N0	Load A	Read A	Zero A	Zero B	Read B	Add'ns	Enable
12A	0	0	X	0	1	1	0	0	0	0	0	0
12B	0	1	X	1	0	0	1	1	1	0	0	0
12C A-max	1	0	0	1	1	0	1	1	1	1	0	0
12D A-min	1	0	1	0	0	0	0	0	1	0	1	1
12E B-min	1	1	0	0	0	0	0	0	1	1	1	1
12F B-max	1	1	1	0	0	0	1	1	0	0	1	1

High-level Design of a State Machine

Construct a synchronous Moore machine with two inputs A and B, and with two outputs X and Y. The machine accepts synchronously with the clock data on the two input lines. The output X is at a logical 1 when the data on the two inputs has been the same ($A=B$) for three or more consecutive clock cycles; otherwise it is at logical 0. Output Y is at logical 1 when the data on the two inputs has been exact complements for three or more cycles.

- A) Give a block diagram of an implementation of such machine using MSI components: Registers, Counters, Flip-flops, Decoders, Multiplexers, Adders, Comparators, etc. Choose a straight-forward approach; no need to go into any details.
- B) Draw a minimum state diagram as a formal problem description for the above machine. Clearly indicate the significance of each state and where the outputs come from.

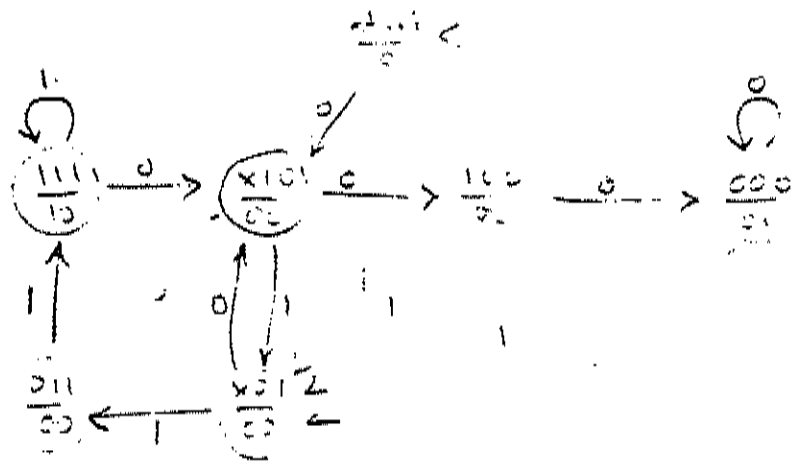
A) straight forward, no FSM per se, just a shift register



B) this is a different design; implementing this would obviate the shift register:

x means either 0 or 1

eg the state named $\begin{pmatrix} x01 \\ 00 \end{pmatrix}$ means $Q_2 = 0$ or 1 (Don't care)
 $Q_1 = 0 \Rightarrow A_{t-1} \neq B_{t-1}$
 $Q_0 = 1 \Rightarrow A_t = B_t$
x output
y output



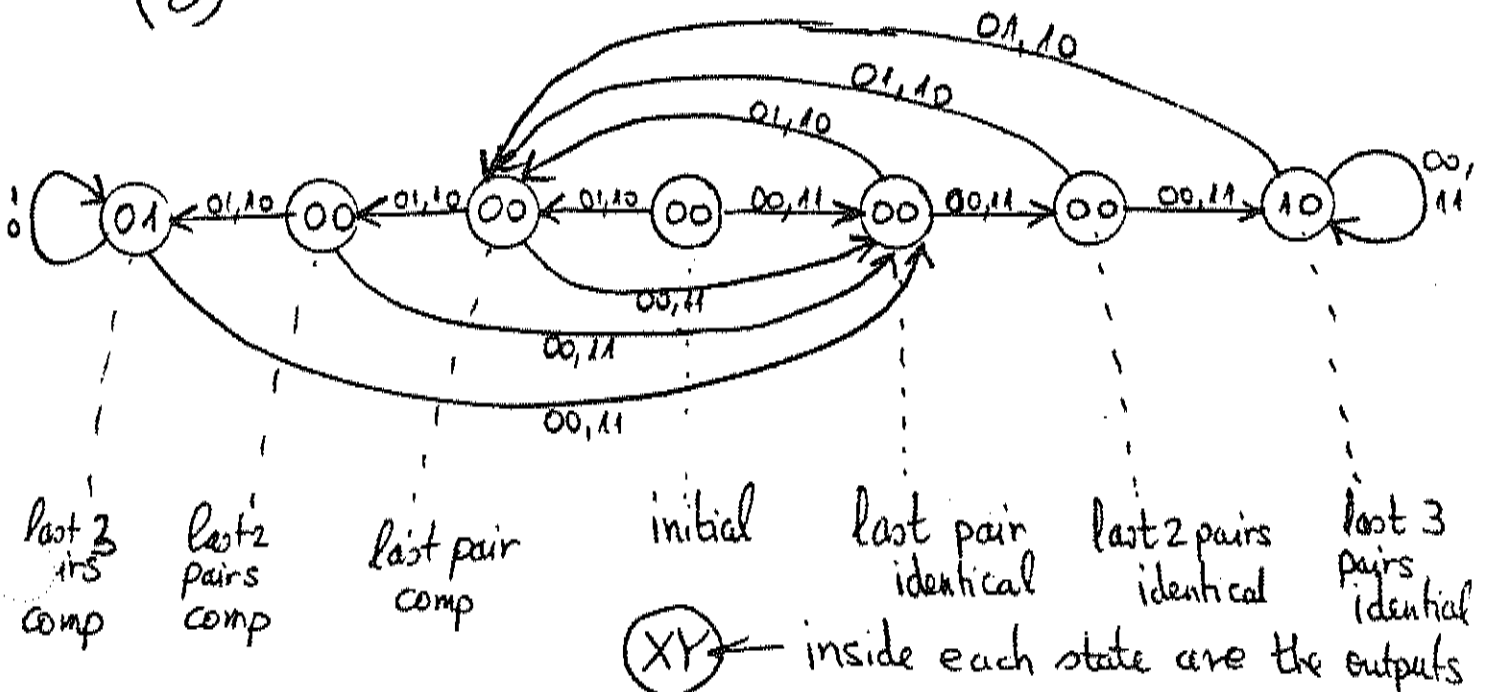
High-level Design of a State Machine

Construct a synchronous-Moore machine with two inputs A and B, and with two outputs X and Y. The machine accepts synchronously with the clock data on the two input lines. The output X is at a logical 1 when the data on the two inputs has been the same ($A=B$) for three or more consecutive clock cycles; otherwise it is at logical 0. Output Y is at logical 1 when the data on the two inputs has been exact complements for three or more cycles.

A) Give a block diagram of an implementation of such machine using MSI components: Registers, Counters, Flip-flops, Decoders, Multiplexers, Adders, Comparators, etc. Choose a straight-forward approach; no need to go into any details.

B) Draw a minimum state diagram as a formal problem description for the above machine. Clearly indicate the significance of each state and where the outputs come from.

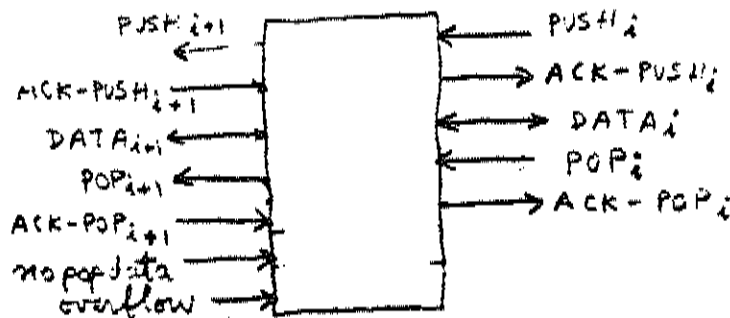
(B) *Label the significance of each state at the bottom!*



10/10

You are to design an asynchronous building block component for a hardware stack. The reusable component implements a single element of the stack. On PUSH, it reads data from the cell on its right and passes its current data to the cell on its left. POP moves data in the opposite direction. By asynchronous, we mean that each element may have its own internal clock, but that the system as a whole has no global clock. Further, PUSH and POP signals cannot be distributed globally.

- 1) Draw a black box diagram of the basic element, indicating all data and control signals it needs.
- 2) Sketch the state diagram from implementing the PUSH function.
- 3) Show a timing diagram for typical PUSH operations.
- 4) Describe how you would deal with an empty stack (underflow) or a full stack (overflow).

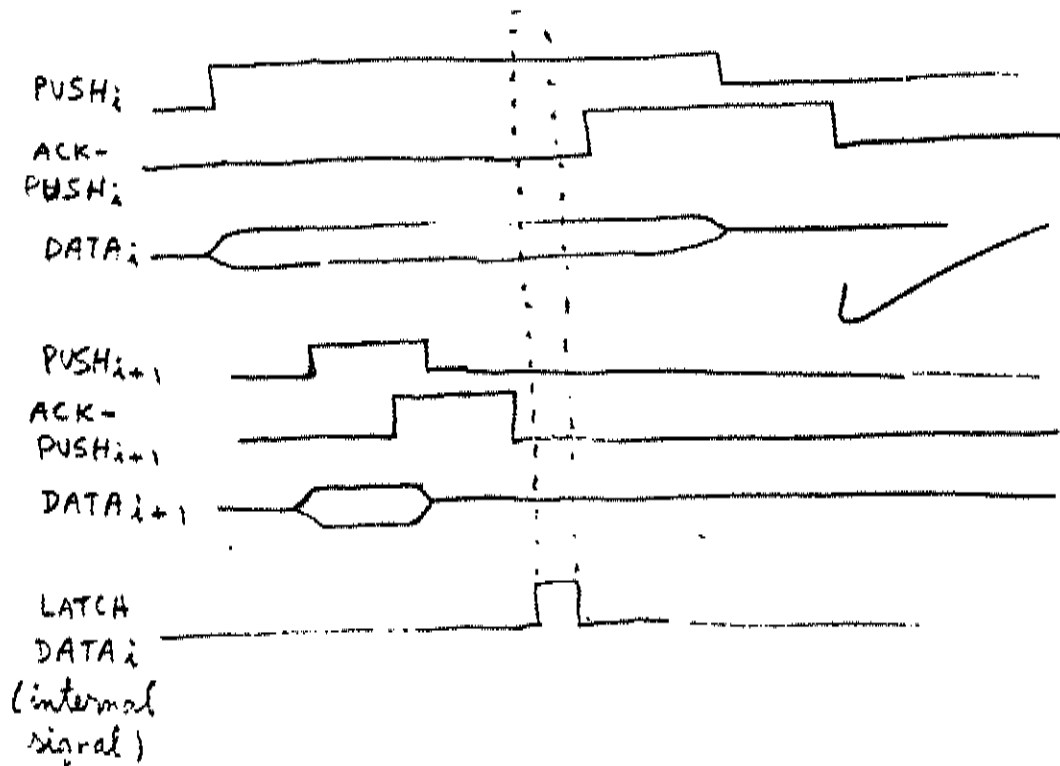
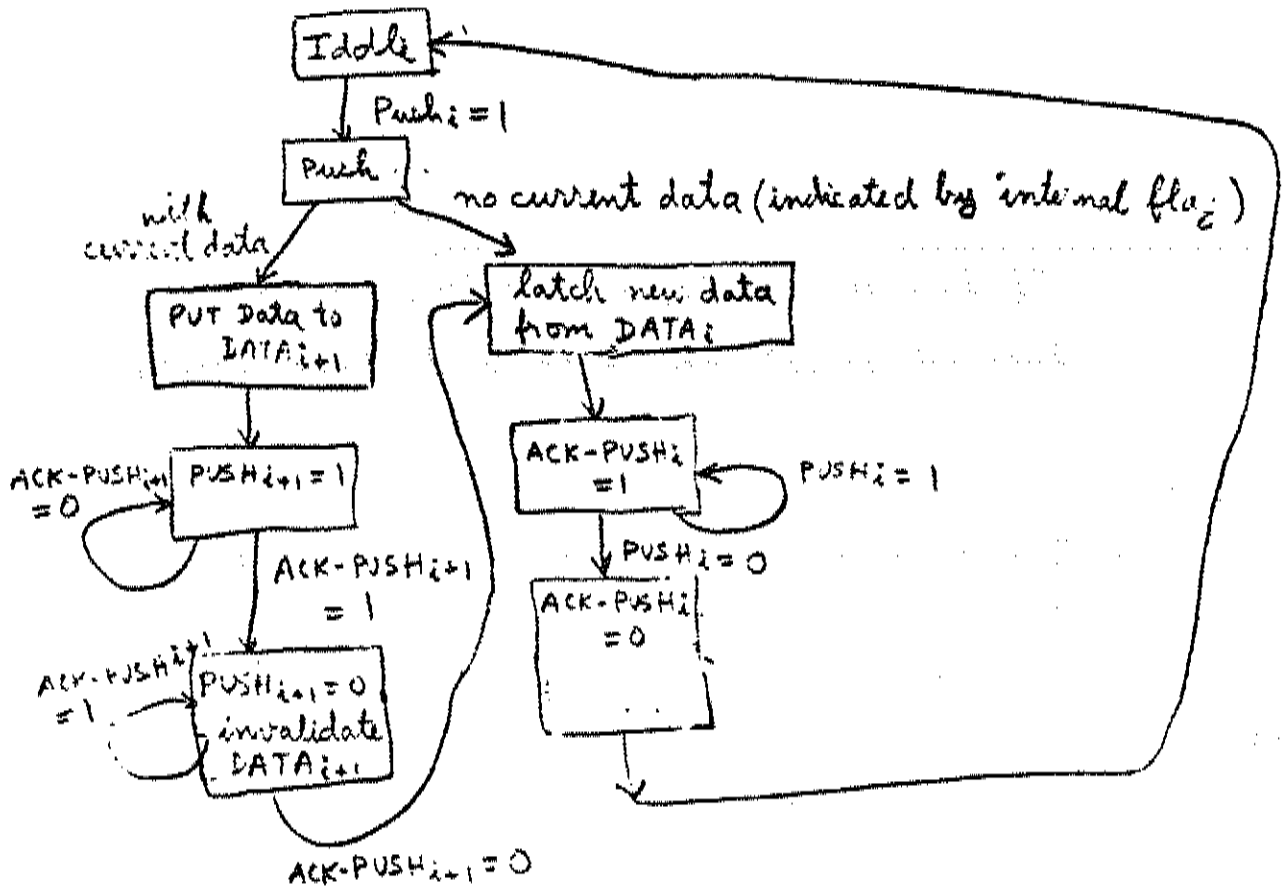


(see other side of page)

- transfer between i^{th} and $i+1^{\text{th}}$ element via 2-wire handshake protocol (described later)
- on push, tries to transfer data to left ($i+1$) before accepting new data from right ($i-1$)
- on pop, tries to transfer data to right ($i-1$) before accepting new data from left ($i+1$)

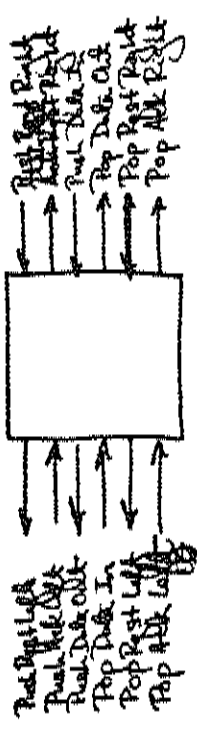
and push is received

- If last one at left has data^v (overflow), it does not accept new data and signal overflow to right (which propagates)
- If last one on right has no data and POP is received, it raises "no pop data" flag to indicate underflow



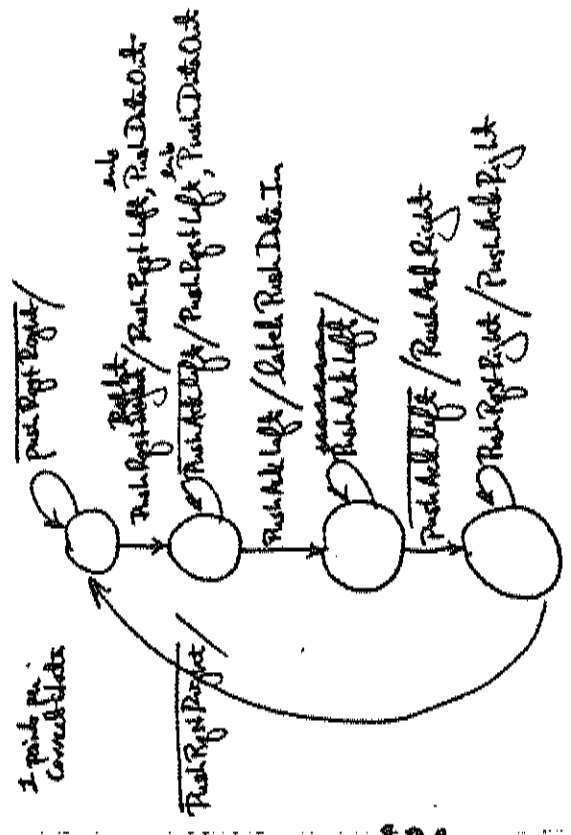
(1) 76 box

← PUSH
POP →

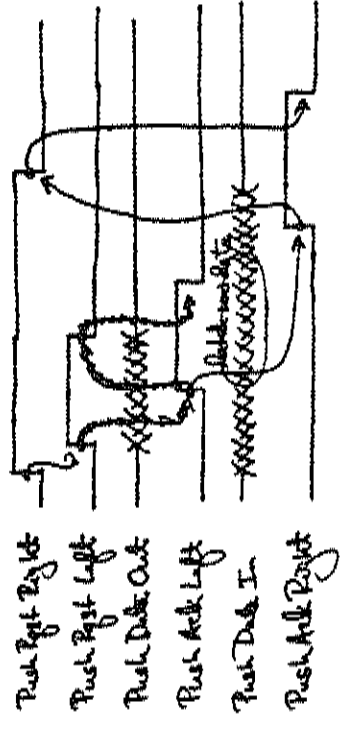


Grading / 4 sets of Push/Ack +2
2 sets of Pop/Ack +1
else 0

(2) Push State Diagram
○ if push/pop signal globally distributed

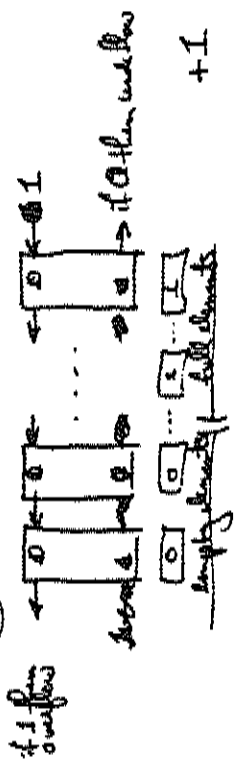


(3) PUSH TD



4 cycle handshake essential! else -1
nested handshake within handshake else -1
some flexibility when rewrite can be tolerated

(4) handling overflow/underflow



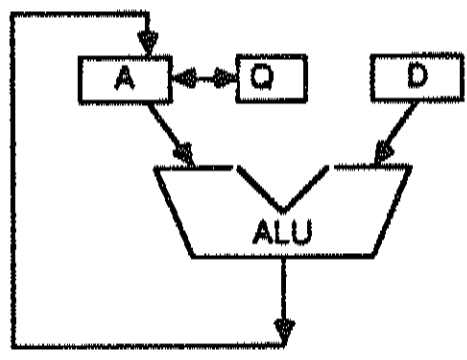
M.S. solution
RHK

10

INTEGER ARITHMETIC

A) What is the key difference between restoring and non-restoring division?

Given this portion of a simple data path and assuming non-negative operands:



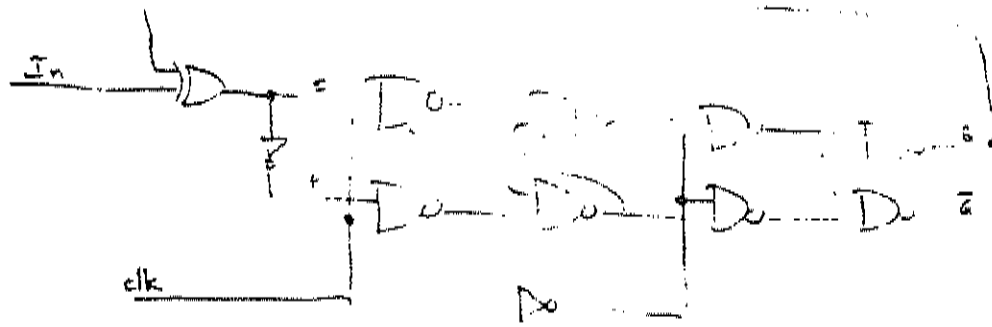
B) Is Restoring or Non-Restoring faster?

C) Give an equation showing how much faster it is.
State your assumptions.

✓ A) The key difference is the way the alg. handles the case when the higher order bits of the dividend is found to be smaller than the divisor after a subtraction is done.
For restoring : the divisor is added back before left shift
non-restoring : left shift is first performed and then the divisor is added using the fact that

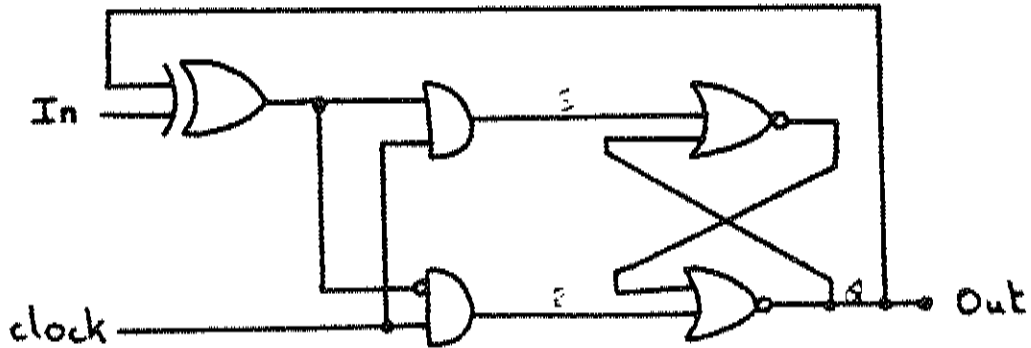
✓ B) Non-restoring is faster because of fewer "+" and "-" operations.
$$2(A-0) < 0 = 2A - B$$

(BEHIND)



10

Given following circuit:



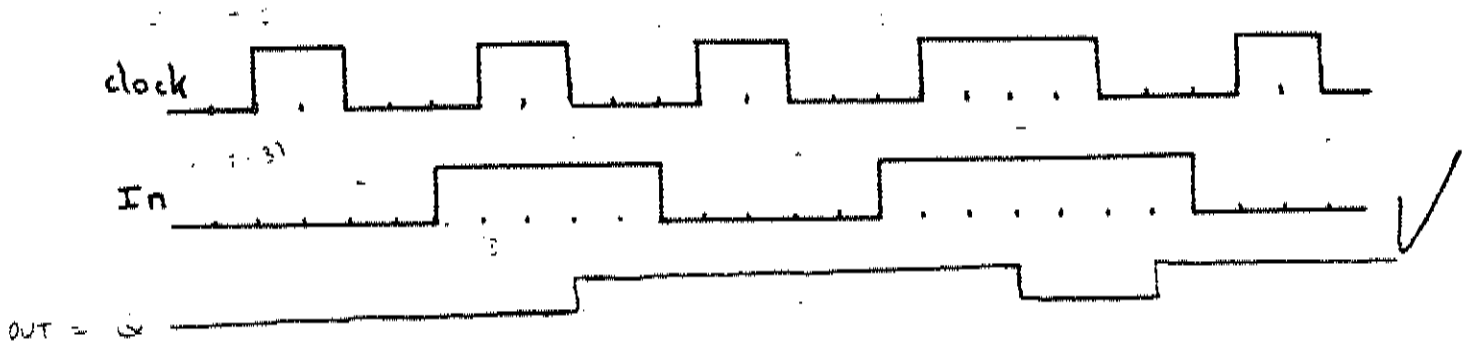
A. What is the function of the above circuit?

*it is: oscillator with end 2
oscillates iff In=1, with period 6* ✓

B. Sketch the output waveform for following clock and In signals:

*it's probably an SR
T flip flop*

Suppose that the delay of all logic elements is identical and equal to one time unit. Assume Out is initially low.



C. Explain the abnormal behavior of the circuit.

How can this be avoided?

Redesign the circuit so that these effects disappear.

This T flip-flop Design is faulty because it employs feedback around a transparent element, a gated SR latch; the changing output influences the input, causing a race condition. ✓

Redesign with a master slave flip flop instead of SR latch.

5 c): Assume the probability of \hat{n} need for restoring is $\frac{1}{2}$.

For operands w/ $2n$ bits

roughly n shifts are needed.

* Ignore the need to restore the remainder: (assume n very big)

Restoring:

No need to restore

$$\frac{n}{2} \text{ times} \times 1 \text{ sub.} = \frac{n}{2}$$

Need to restore

$$\frac{n}{2} \text{ times} \times (1 \text{ sub} + 1 \text{ add}) = n$$

$$\text{Total} = \frac{3n}{2}$$

Non-restoring:

$$n \text{ times} \times 1 \text{ (add/sub)} = n$$

\therefore Non-restoring is about $\frac{\frac{3n}{2}}{n} = \frac{3}{2}$ times faster.

10

MICROPROGRAMMING

✓ A) What is the key reason for using a two level control store?

To reduce the size of the control store, by allowing ~~micro~~ instructions shared by many sequences

to be stored only once in the second-level control store. The width of the first level, and the depth of the second level, are smaller than the corresponding

B) Suppose you are building a microprogrammed computer and the only ^{addresses of the control} memory chips available are 4096 word by 1 bit chips. A trial one level design requires 4096 by 100 bits (100 chips). Given this situation, in what circumstances (if any) would a two-level control store design be superior?

✓ Since the width of the μ -instruction is 100 bits, the width of the second-level store would also have to be 100 bits, making it as large as your original first-level store. This is a bad idea.

It is also pointless to try to decrease the number of addresses by using a 2-level store, since the original microprogram fits in the one-level store.

So it would not be desirable.

EXCELLENT ANSWERS

~~20~~
20

Given only two component types:

- A. 2-input NOR gates,
- B. tri-state bus drivers,

but as many as you like, show a hardware design for a 32-Bit Processor with following instruction set:

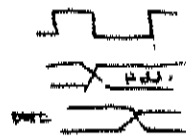
```

sub (x): AC ← AC - m(x)
stor (x) m(x) ← AC
branch (x): IF AC ≥ 0 THEN PC ← X.
    
```

A large part of your score for this problem will depend upon your organization and neatness in presenting a solution. Use hierarchical descriptions. Use good engineering design practice in your decisions. If an assumption is needed, make the simplest assumption and state clearly what it is.

AGAIN: BE WELL ORGANIZED AND NEAT!

Assume: Memory cycle complete in $\frac{1}{2}$ cycle
clock



data can be latched by edge-triggered F/F here.

∴ only use edge-triggered F/F for simplicity.

Inst. Format

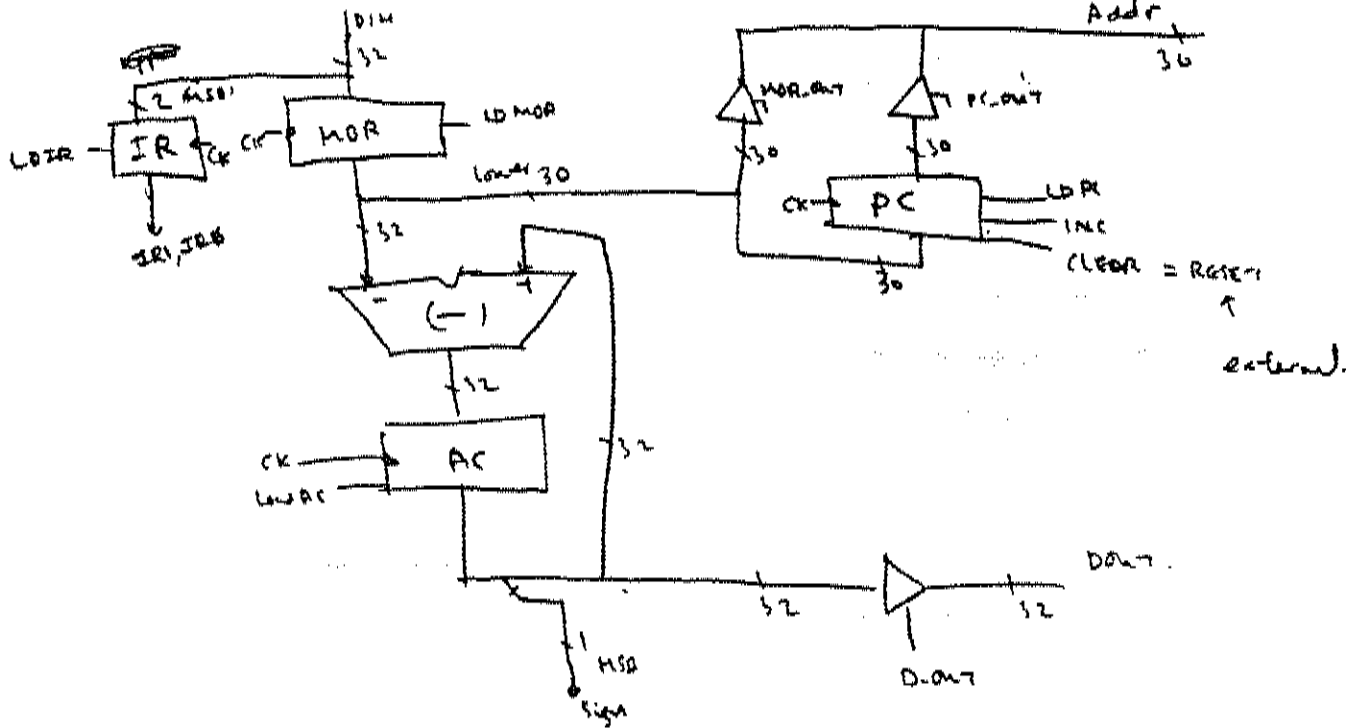
- 00: sub X
- 01: stor X
- 11: Brn X



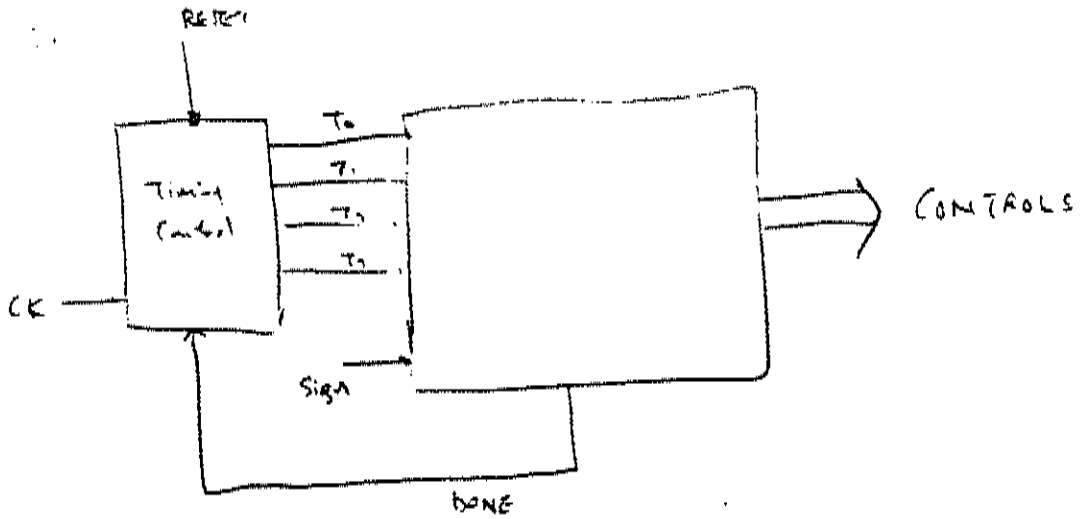
Problem 7
50 Minutes

ID# 100

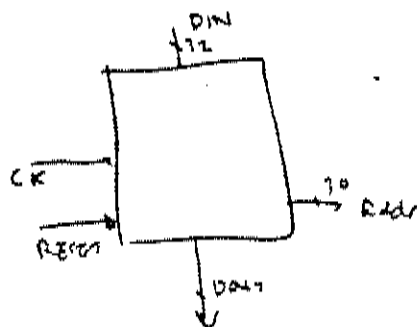
Register Diagram of E-unit



Control Unit

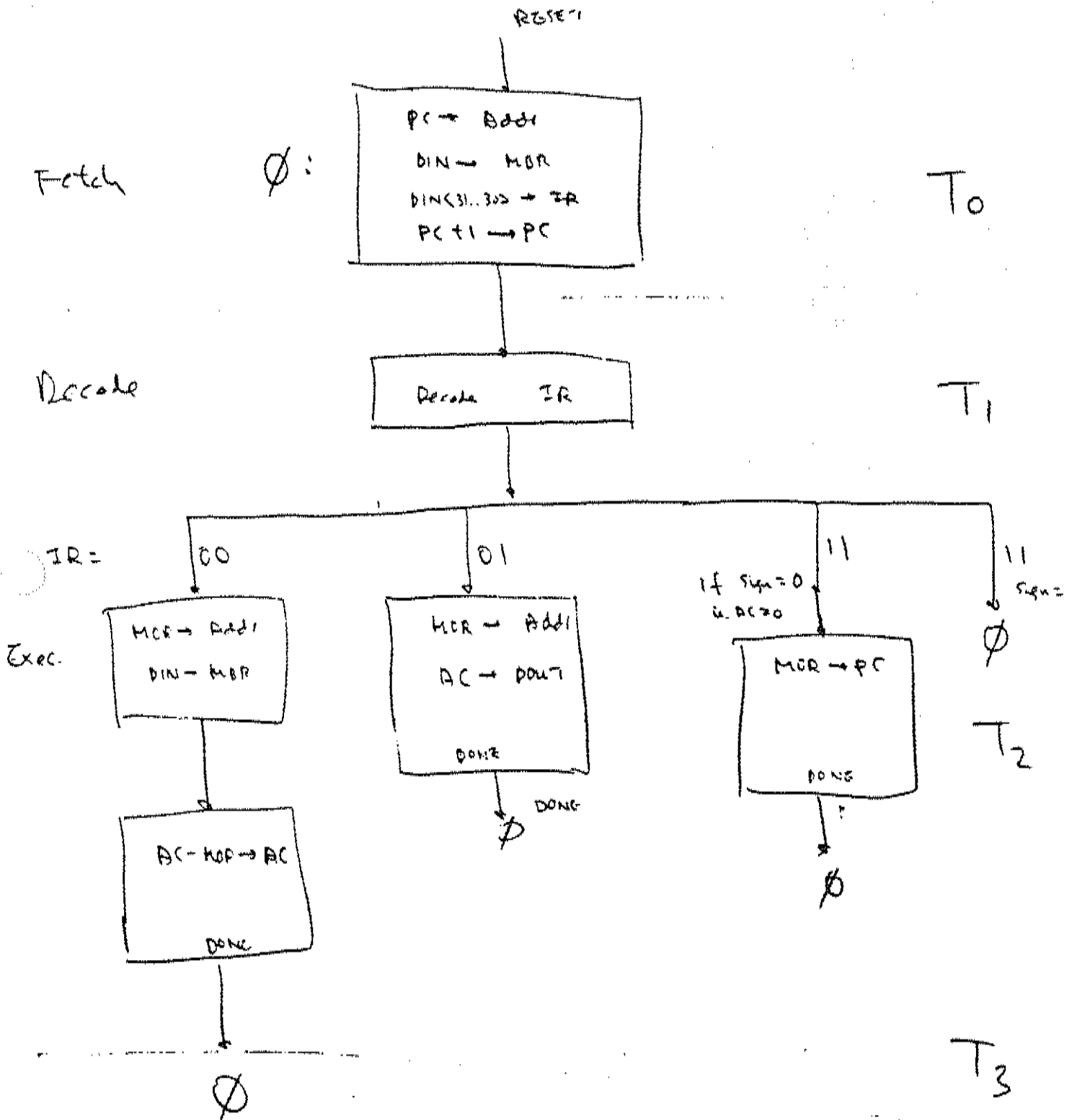


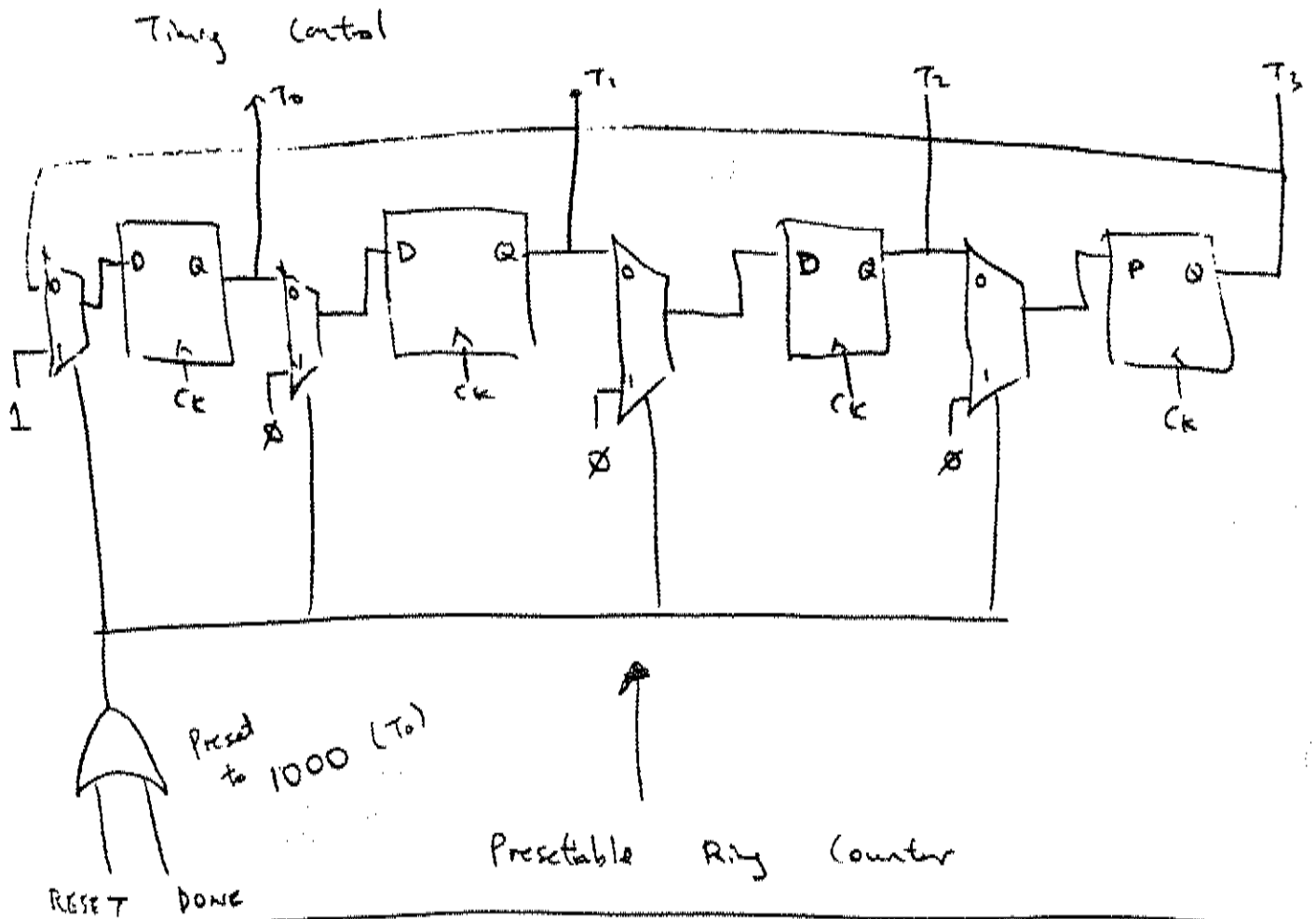
Top



Upon reset
 $PC = 0$
and starts fetching
instruction

State Diag



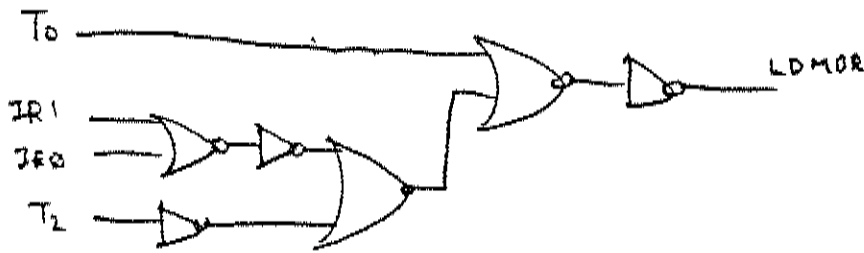


Control Truth Table

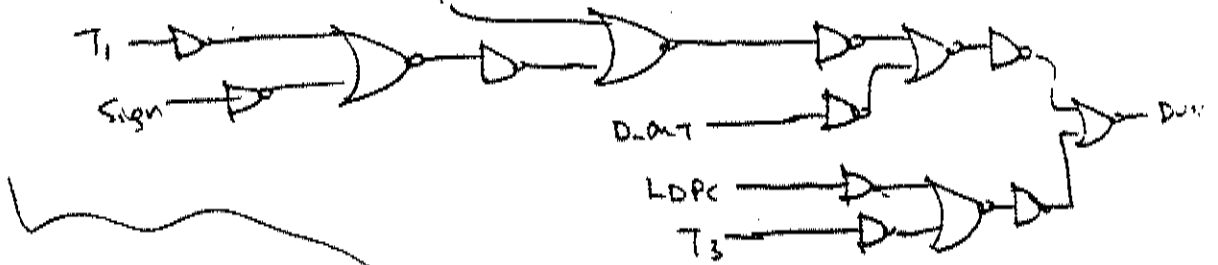
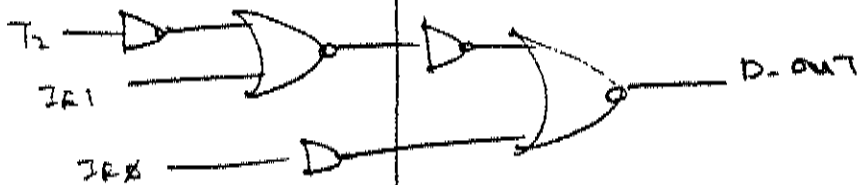
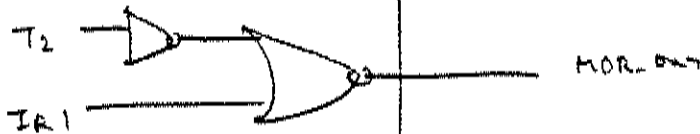
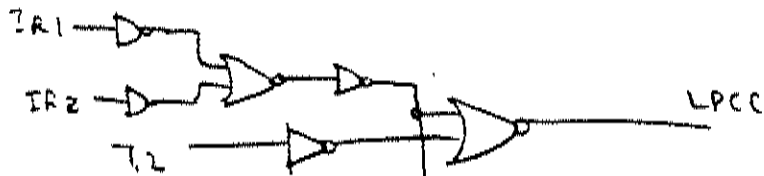
Timing	Sign	ER	LDRA	LDRA	LDRC	LDPC	INCR	MRDN	PC_OUT	E_LMT	DONE
T ₀	X	XX	1	1	0	0	1	0	1	0	0
T ₁	1	11	0	0	0	0	0	0	0	0	1
T ₁	(^{next} 1111)		0	0	0	0	0	0	0	0	0
T ₂	X	00	0	1	0	0	0	1	0	0	0
T ₂	X	01	0	0	0	0	0	1	0	1	1
T ₂	X	11	0	0	0	1	0	0	0	0	1
T ₃	X	XX	0	0	1	0	0	0	0	0	1

Implementation

$T_0 \rightarrow \text{LDIR, INCPC, PC_OUT}$



$T_3 \rightarrow \text{LDAC}$



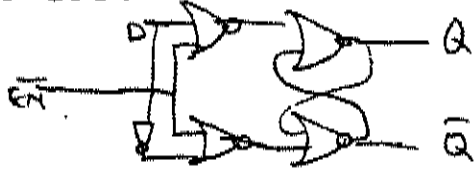
Should do better with active low signals instead.

Too many levels: but no time to optimize.

Logic Composites

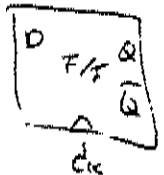
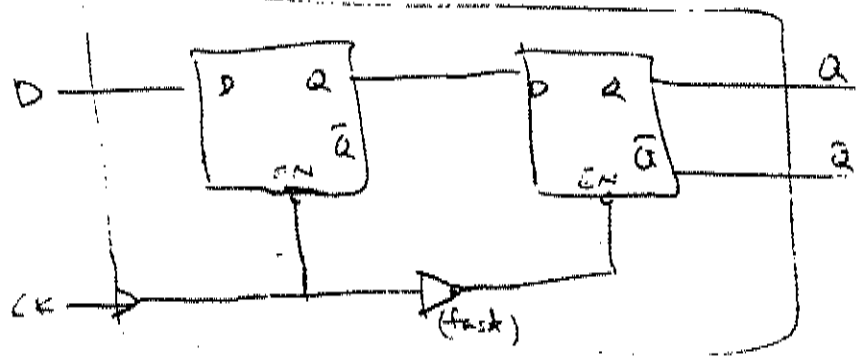


D- Latch

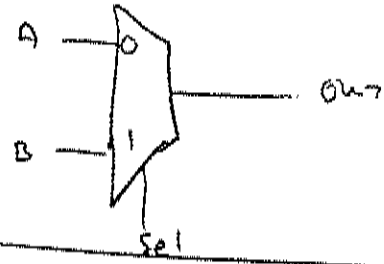
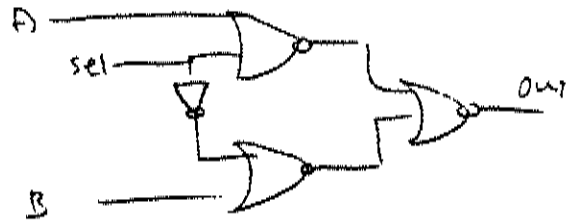


D F/F

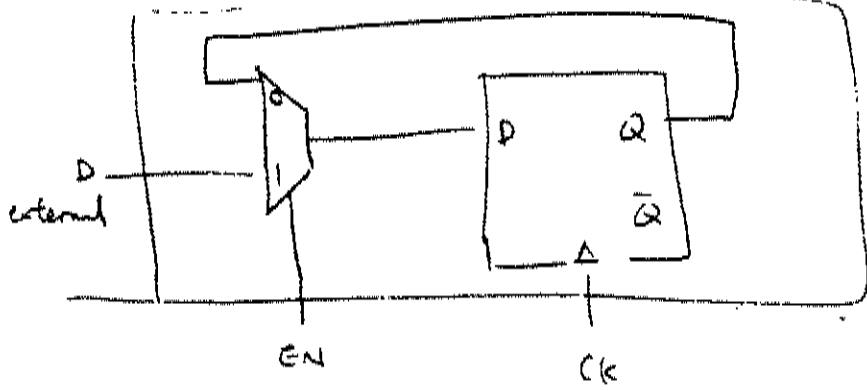
edge trigger



2 to 1 Mux



D F/F w enable



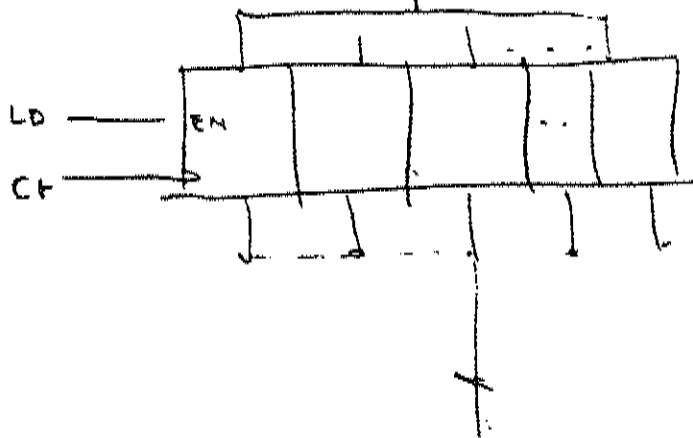
P.11
Pg 7.

#66

For AC, MBR, IR,

32 for R
32 " MBR
2 for IR

LEDC/
LDMR/
LDTR



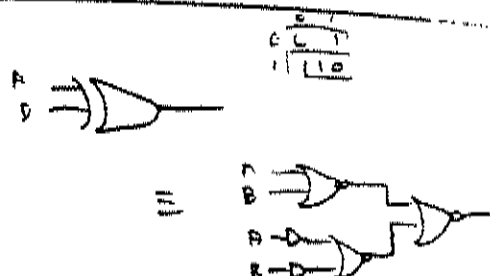
1 bit full adder

$$S = A_i \oplus B_i \oplus C_m$$

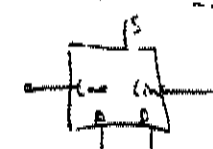
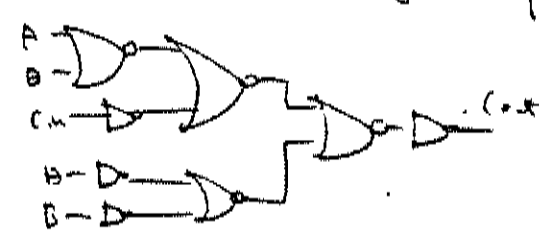


$$C_{out} = AB + AC_m + BC_m$$

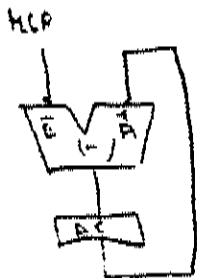
$$= AB + (A+B)C_m$$



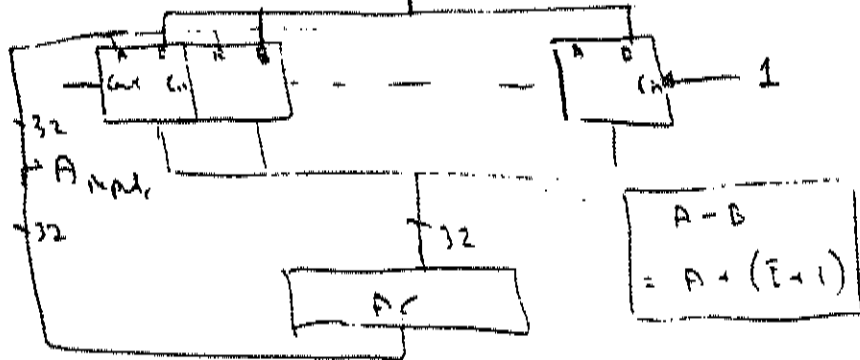
What maximum
depth?



Too many levels
No time to optimize.

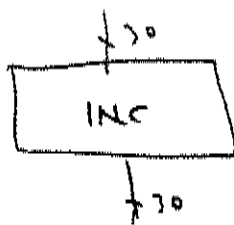


ABR takes 2 outputs,
f₁₂

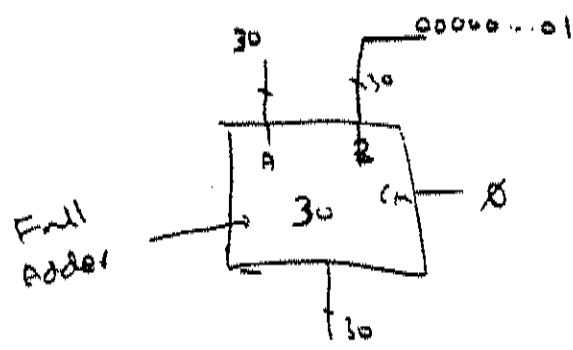


PC

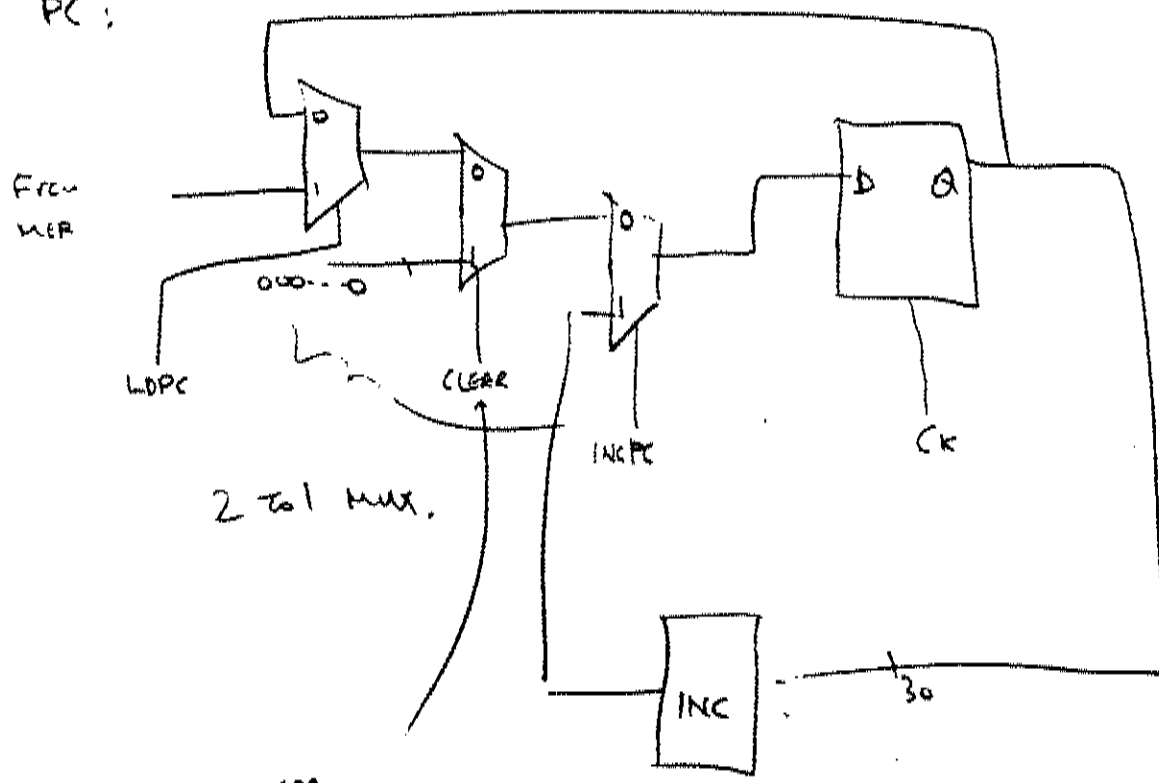
INC.



Use 30 Full adders



PC :



CLEAR = RESET

HARDWARE CORE EXAM
FALL 1988

Your number: _____

Score: _____

Hardware Preliminary Examination

19 September 1988

INSTRUCTIONS

1. This is a closed everything exam. Do not have anything near you except this examination, pencils (or pens), rulers, templates or erasers. (Calculators, scratch paper, etc. may not be brought to the examination.) Use the back pages of the exam for scratch paper. If you need more scratch paper, it must be obtained from the exam proctor. Try to leave an empty seat between you and your neighbors.
2. The exam will last exactly 3 hours (180 min.). Late exams will not be accepted.
3. The exam has a total of 180 points (~ one point per estimated minute to be spent on each problem).
4. If you have a question, raise your hand. Check the blackboard for exam updates, typo fixes, etc.
5. Be sure to put your answers inside the boxes provided for that purpose.
6. Write neatly. Be well organized.
7. State all assumptions.
8. Try to do the easy (for you) problems first.
9. Show your work. In the case of an incorrect answer partial credit may be given.

Note: The notations x' and \overline{x} are equivalent.

a) State DeMorgan's Theorem for two variables A and B:

b) Prove a).

c) Simplify the following using algebraic operations:
Show your steps.

i) $(A + B)(A + C)$

answer

ii) $(AB' + A'B + A'B')$

answer

d) Use a Karnaugh map to simplify the following:

$$ABCD + A'B'C'D' + A'BC'D' + A'B'C'D + A'BC'D + ABCD' + AB'CD' + AB'CD$$

		AB			
		$\bar{A}\bar{B}$	$\bar{A}B$	$A\bar{B}$	AB
$\bar{C}\bar{D}$	1				1
$\bar{C}D$	1				1
$C\bar{D}$		1	1		
CD		1	1		

answer

$$\bar{A}\bar{C} + AC$$

e) Given the following truth table with d representing the "don't care" condition, write a simplified function using a Karnaugh map:

A	B	C	
0	0	0	d
0	0	1	d
0	1	0	d
0	1	1	d
1	0	0	1
1	0	1	d
1	1	0	0
1	1	1	0

		AB			
		$\bar{A}\bar{B}$	$\bar{A}B$	$A\bar{B}$	AB
\bar{C}	0	d	d	d	d
C	1	1	d	d	d

answer

$$B$$

Problem #1

f) Given the following truth table:

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

$F = \bar{A}\bar{B}C + \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}\bar{C} + AB\bar{C}$
 $F = \bar{A}(\bar{B}C + B\bar{C} + BC) + A(\bar{B}\bar{C} + B\bar{C})$
 $F = \bar{A}(\bar{B} + B)(C + \bar{C}) + A(\bar{B} + B)(\bar{C} + C)$

Write F in:

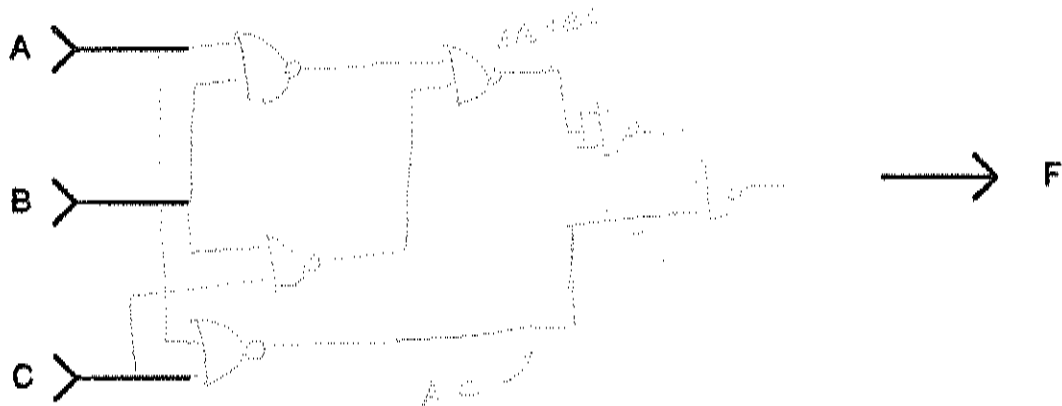
i) sum of products form:

answer

ii) product of sums form:

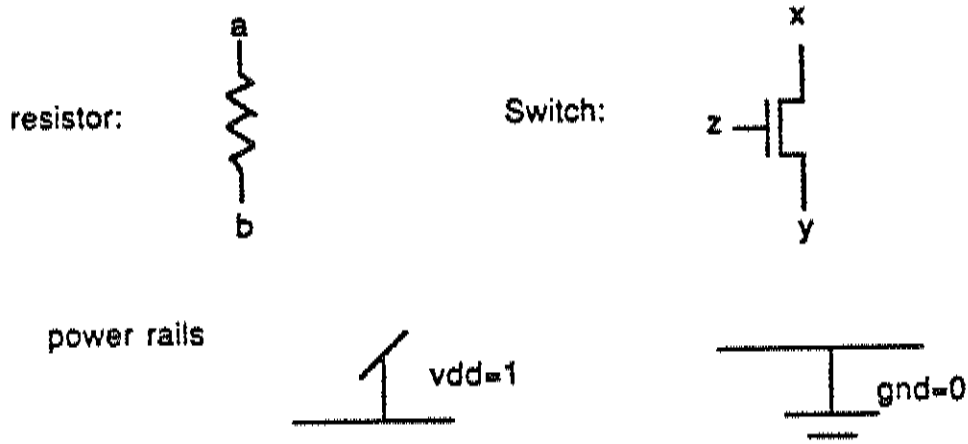
answer

g) Draw a circuit using only 2-input NAND gates to implement $AB + BC + AC$:



Problem #2 (15 points)

- a) You are given two circuit elements, a resistor and a switch, and two power rails, VDD and GND called 1 and 0, respectively.

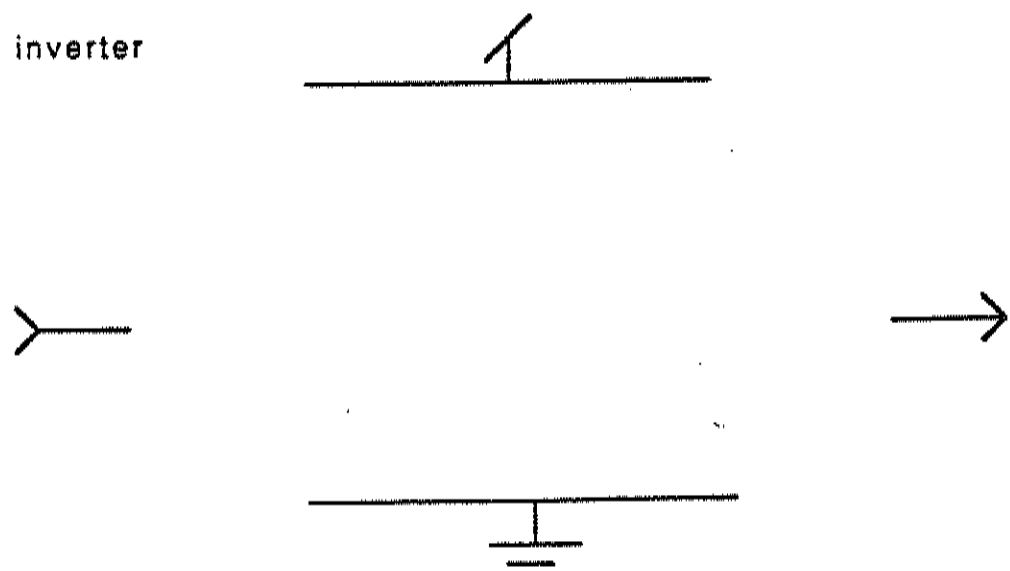


The circuit elements function as follows:

Switch: if terminal z is connected to 1 then x is connected to y.

Resistor: if terminal a is connected to 1 (or 0) then terminal b is forced to 1 (or 0). The switch is stronger than the resistor & wins in any conflicts.

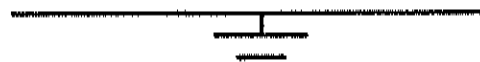
Show a circuit for each of the following. You may use as many elements as needed, but try to minimize the number of elements in each circuit.



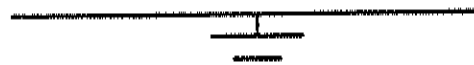
2- input
NAND gate



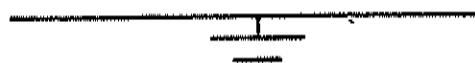
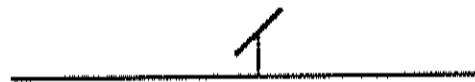
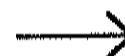
NOR gate
2-input



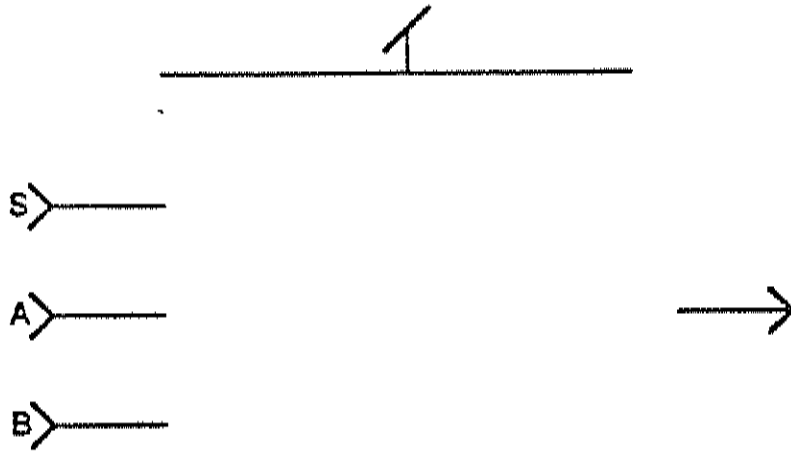
2-input
AND gate



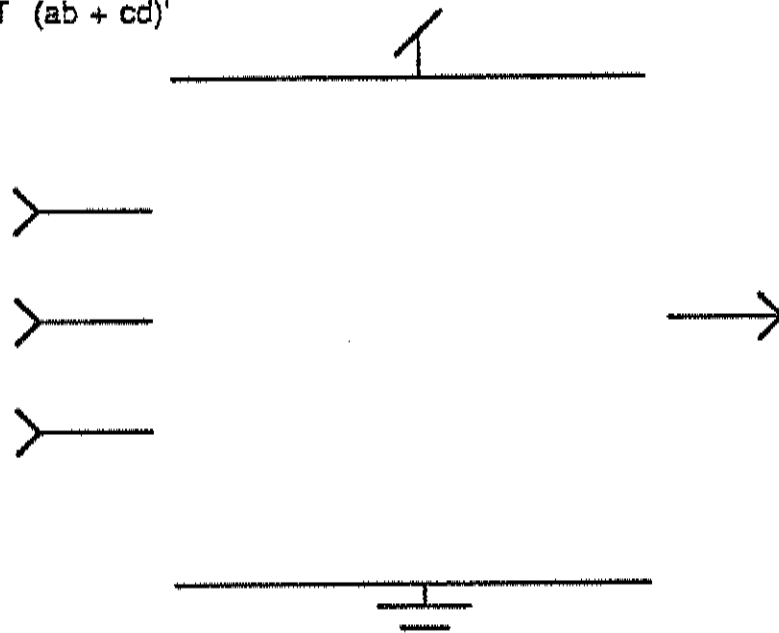
2-input
exclusive-NOR
gate



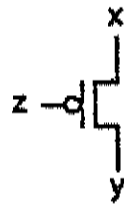
2 x 1
multiplexer



AND/OR/NOT ($ab + cd$)'
gate



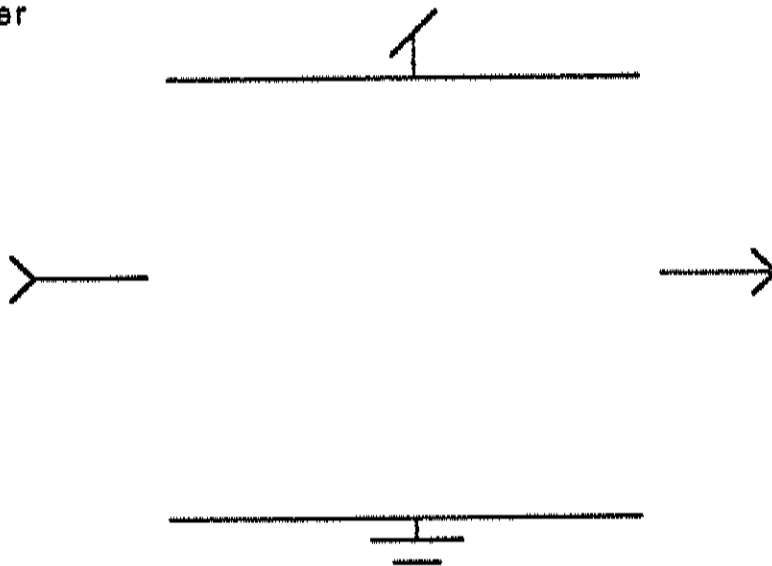
b) Now assume that instead of a resistor, you are given another switch with the following function:



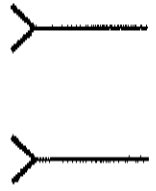
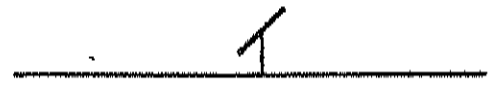
if terminal Z is connected to 0 then X is connected to Y

Show a minimal circuit for each of the following. Again, try to minimize the number of elements in each circuit.:

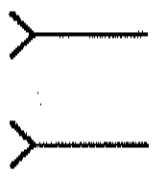
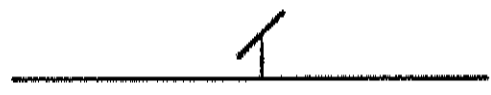
Inverter



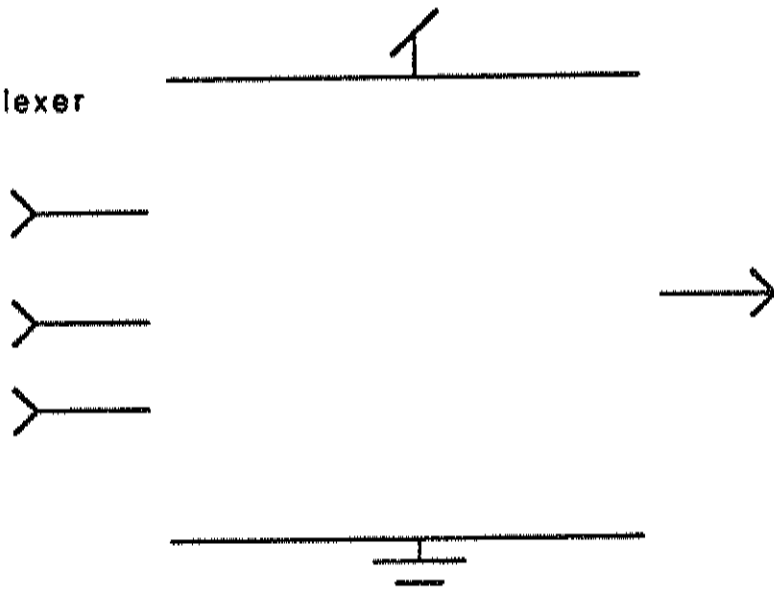
2-input
NAND gate



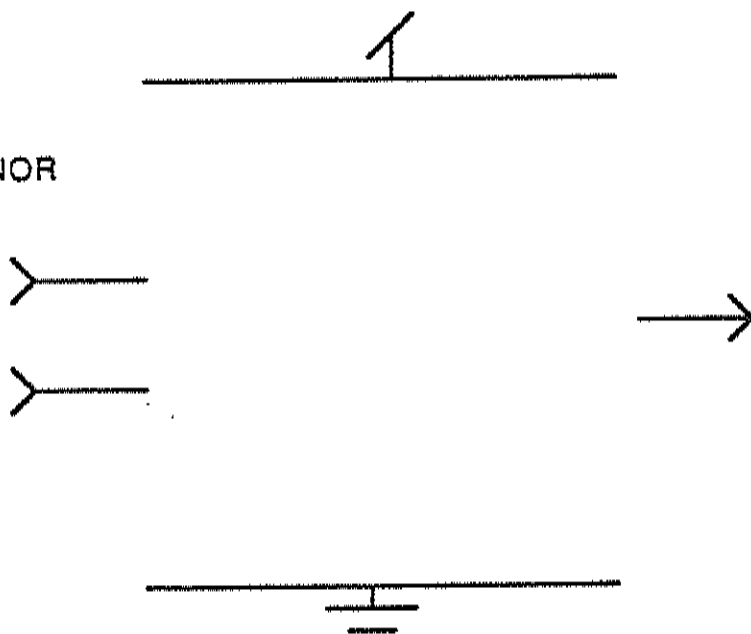
D-type
flip/flop



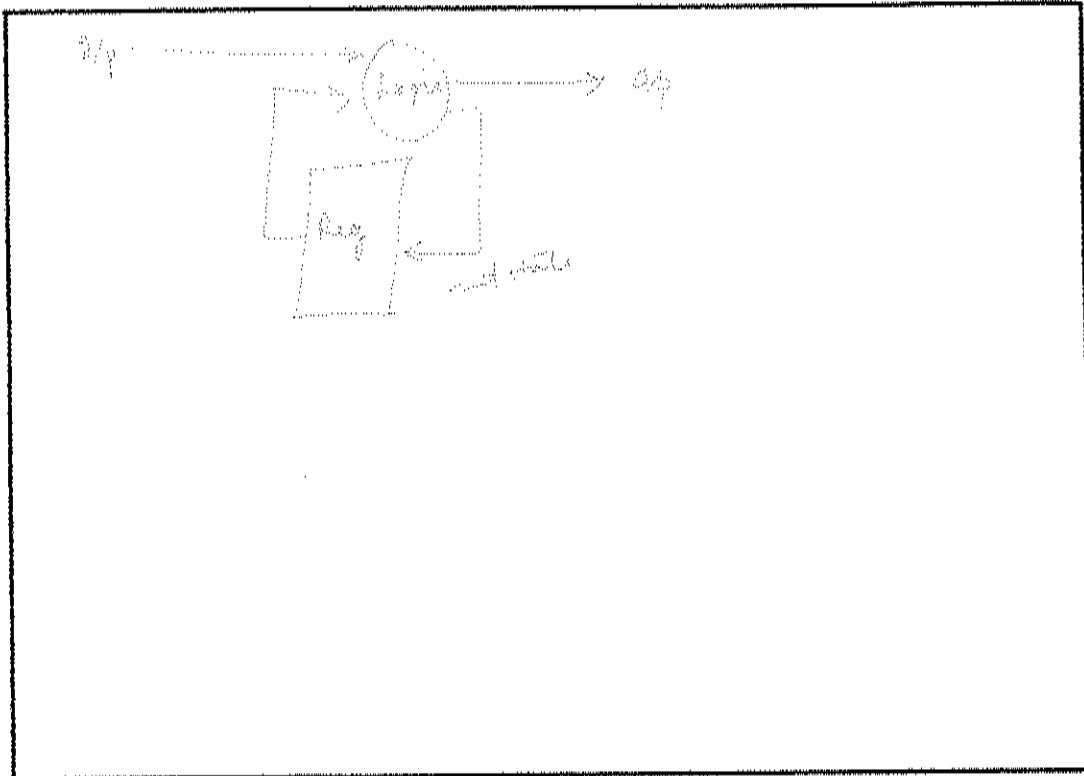
2 x 1
multiplexer



2-input
exclusive-NOR
gate



A. Show a register-level diagram of a Mealy FSM (Finite State Machine).



B. When should a Mealy machine be used?

Blank area for the answer to question B.

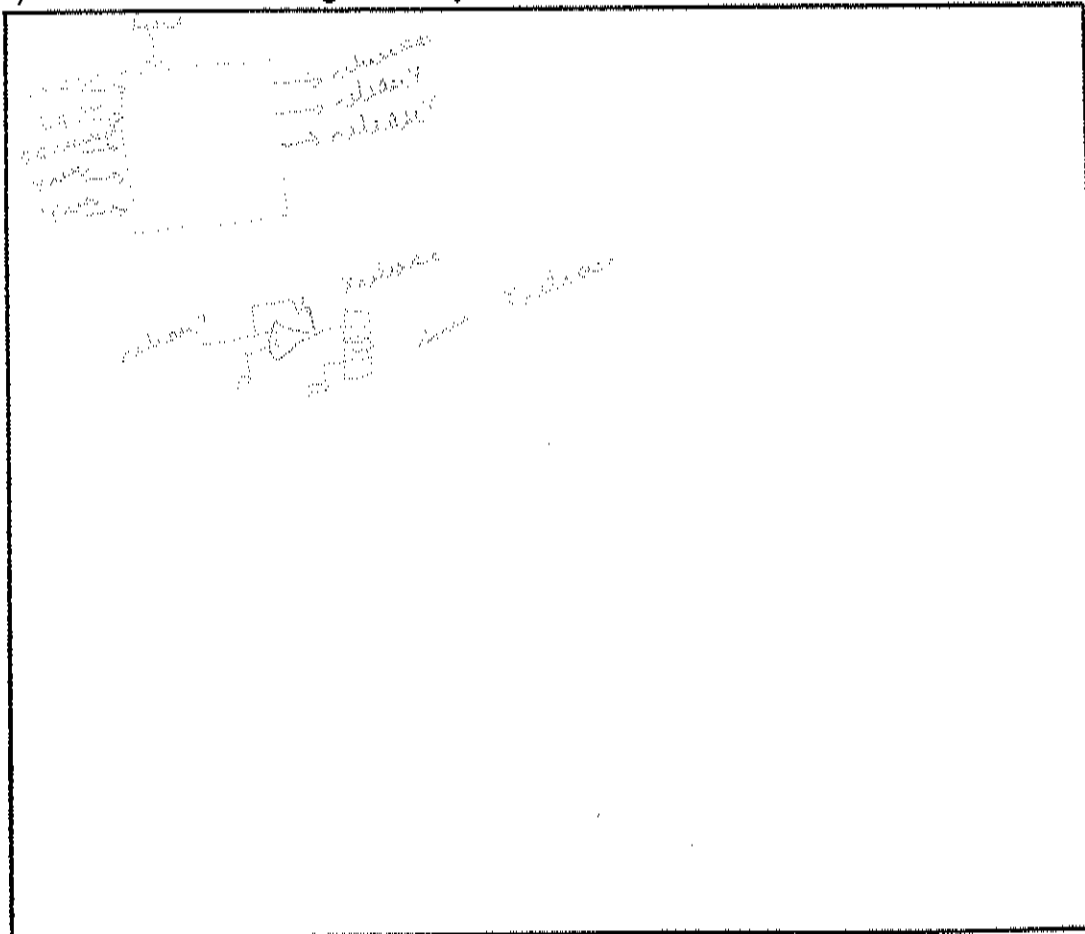
C. Contrast the relative advantages and disadvantages of a Mealy vs. a Moore machine.

ADVANTAGES	DISADVANTAGES

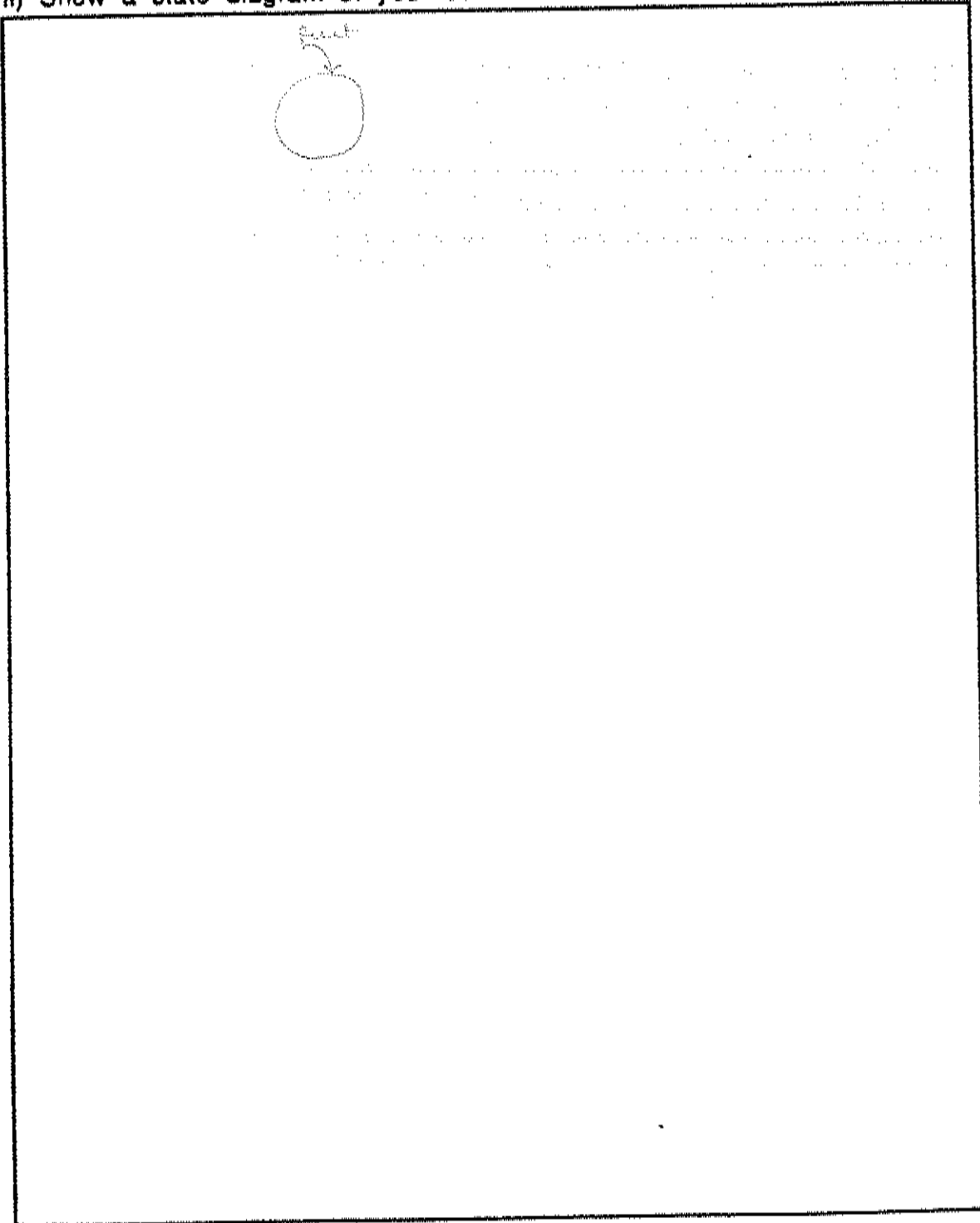
D. A vending machine accepts nickels and quarters and dispenses two products X & Y and change in nickels. X costs 15c and Y costs 10c.

There are five switches. Two sense the passage of quarters and nickels to the coin box. Two are push buttons the customer uses to select the desired product. One senses the passage of each nickel dispensed as change. There are three magnets. One to release nickels for change and two to release products X & Y. There are two power supplies: +5v.dc and a 12 volts AC at 60Hz. Design a complete circuit (controller) to control the vending machine. You may employ diodes, master-slave flip-flops, power operational amplifiers, gates and PLA's. You are required to use a Moore FSM for this controller

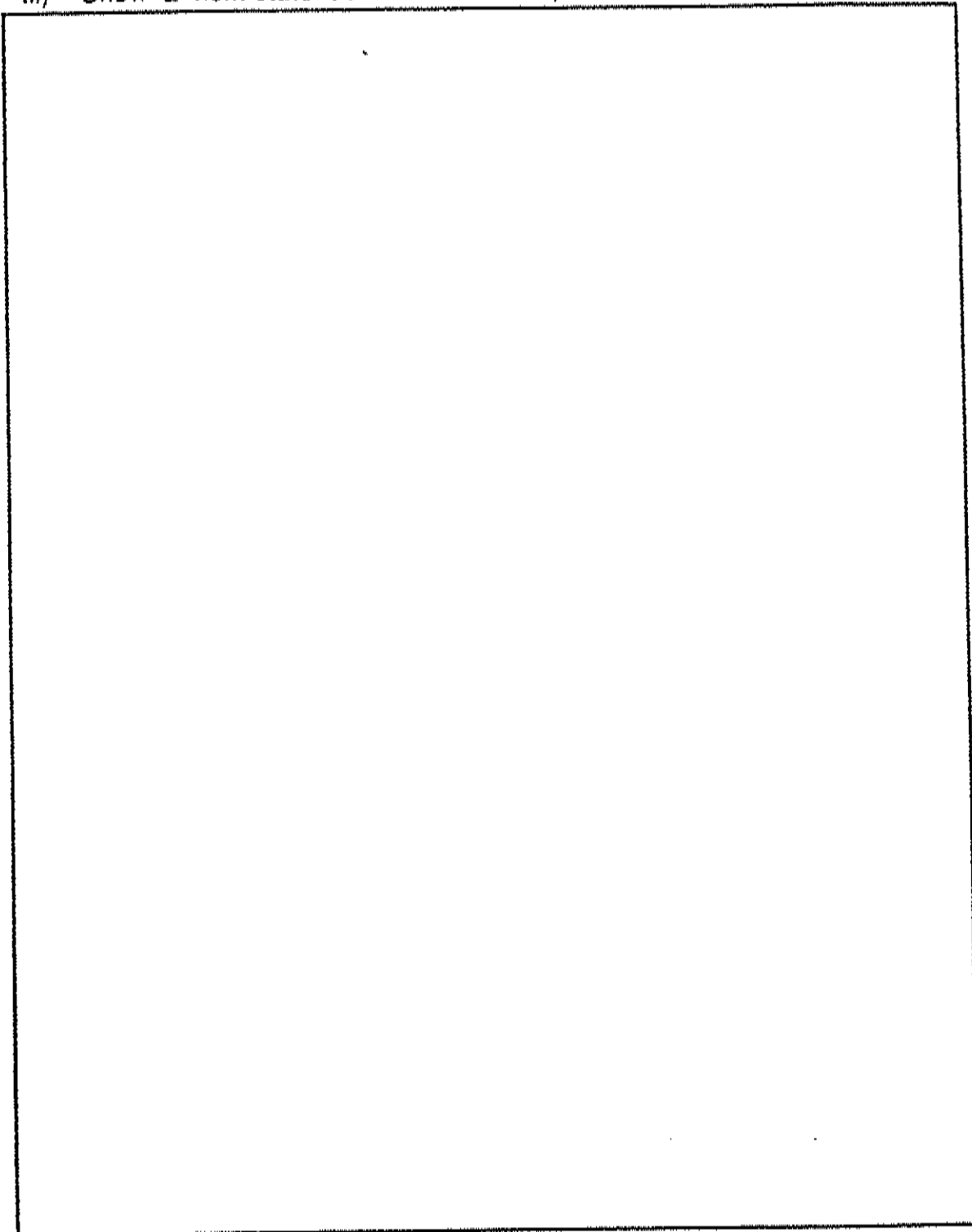
i) Show a block diagram of your machine



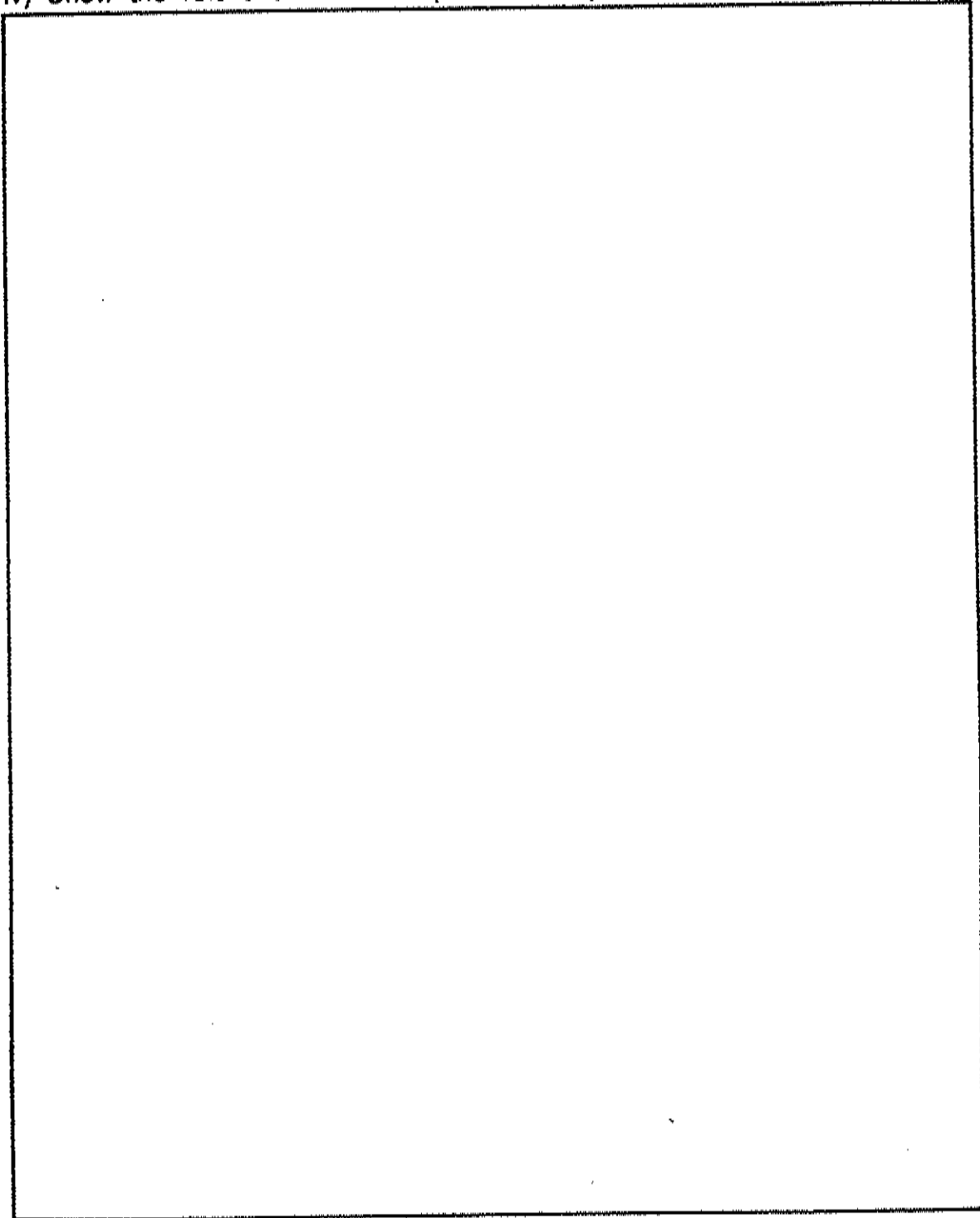
ii) Show a state diagram of your controller.



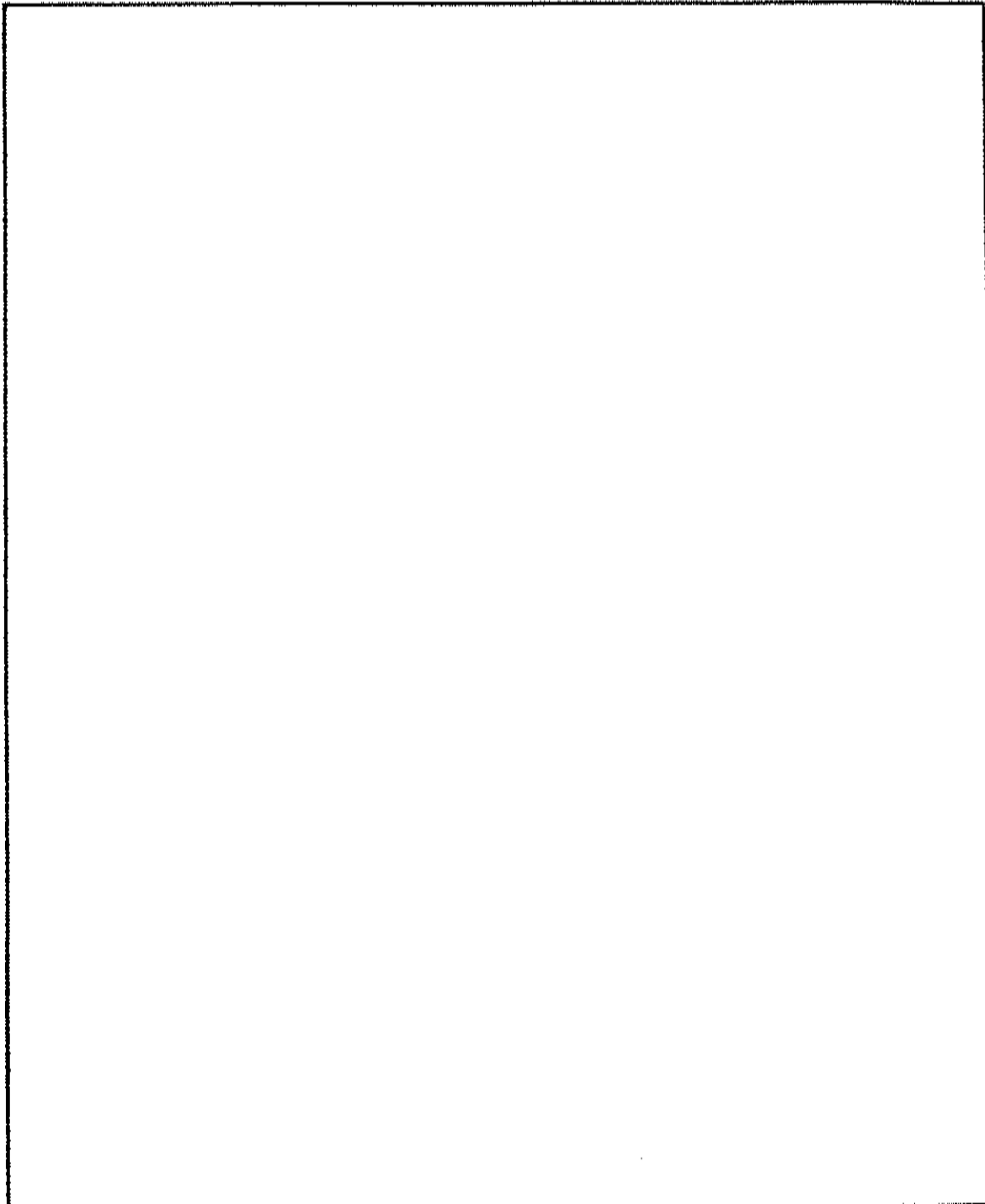
iii) Show a next-state transition and output truth table.



iv) Show the relevant Boolean equations for your controller.

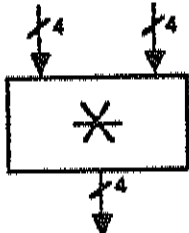
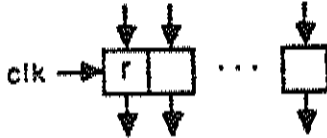

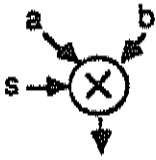



- v) Show the circuit diagram of your controller. Be sure to use a PLA to implement the main FSM.



Problem #4 (30 points)

You are given four 4-bit numbers in read-only registers x1, x2, x3, and x4, along with a supply of parts. Each part has a price in dollars (\$) and a propagation delay in nanoseconds (ns). You may use as many of each part as you need. The four numbers are in unsigned fractional representation

Part	Symbol	Price	Delay	Notes
multiplier		6	10	All normalization & rounding is done within the multiplier.
read/write register bits		0.1 per bit	1	clk is a load signal. You do not need to show a circuit to drive clk.
counter bits		0.1 per bit	1 per bit	clk is a clock signal, as with the register you do not need to show a circuit to drive clk. The total delay for a counter is 1ns X # of bits.
2x1 Multiplexer		0.1	1	s is the select input, you do not need to show a circuit to generate s.
inverter		0.1	1	

You can assume that the register & counter bits start off initialized to all zeros.

- a) Draw a circuit that will multiply the four numbers in minimal time. The result need not be latched. What is the time if the circuit were extended to handle N 4-bit numbers? What is the price in each case?

Price for N :

Price for four:

Time for four:

Time for N :

- b) Draw a circuit that will multiply the four numbers at a minimal price. Show the prices & times for the four numbers and if the circuit were extended to handle N 4-bit numbers.

Price for four:

Price for N:

Time for four:

Time for N:

c) Now assume that each nanosecond of delay costs \$.05. What is the combined cost (cost due to price PLUS cost due to delay time) of your answer in a) & b) for four numbers?

combined cost of a):

combined cost of b):

Is there a circuit for four numbers that has less cost, under this measure?

Draw the circuit:

combined cost:

Problem #5 (20 points)

a) Convert the following from decimal to 8-bit two's complement form:

$-75_{10} =$

b) Convert the following from two's complement to decimal:

$11100001_2 =$

c) Convert from decimal to two's complement fixed point:

$0.65625_{10} =$

Problem #5

- d) Multiply the following two's complement integers.
Show all partial products. Show an 8-bit result:

$$\begin{array}{r} 1011 \\ \times 1001 \\ \hline \end{array}$$

answer

171

- e) Consider two 4-bit two's complement fractional numbers in the following form: S. X X X (sign-bit, followed by the binary point, followed by three fraction bits), multiply the two numbers. Show all partial products. Express your result in the same form, i.e. S. X X X, using

i) truncation:

ii) rounding:

$$\begin{array}{r} 0.010 \\ X \underline{1.011} \end{array}$$

f) Consider two normalized floating-point number systems, NS1 and NS2.

NS1 has four bits of unsigned fractional mantissa in radix-2, and two bits of unsigned exponent.

NS2 has four bits of unsigned fractional mantissa organized as two radix-4 digits, and two bits of unsigned exponent.

Fill in the following table for each system:

	NS 1	NS 2
smallest mantissa		
largest mantissa		
largest exponent		
largest representable value		
number of fractions		
number of exponents		
number of values		

Problem #6 (55 points)

You are to design a computer. It is:

- a) a very simple, von Neumann machine
- b) fully synchronous, with single cycle memory
- c) 16 bits per word
- d) to be constructed of ONLY NAND gates, of arbitrary fan-in and fan-out.
- e) 2's complement arithmetic

You are to describe your design 'top-down'.

The complete instruction set is:

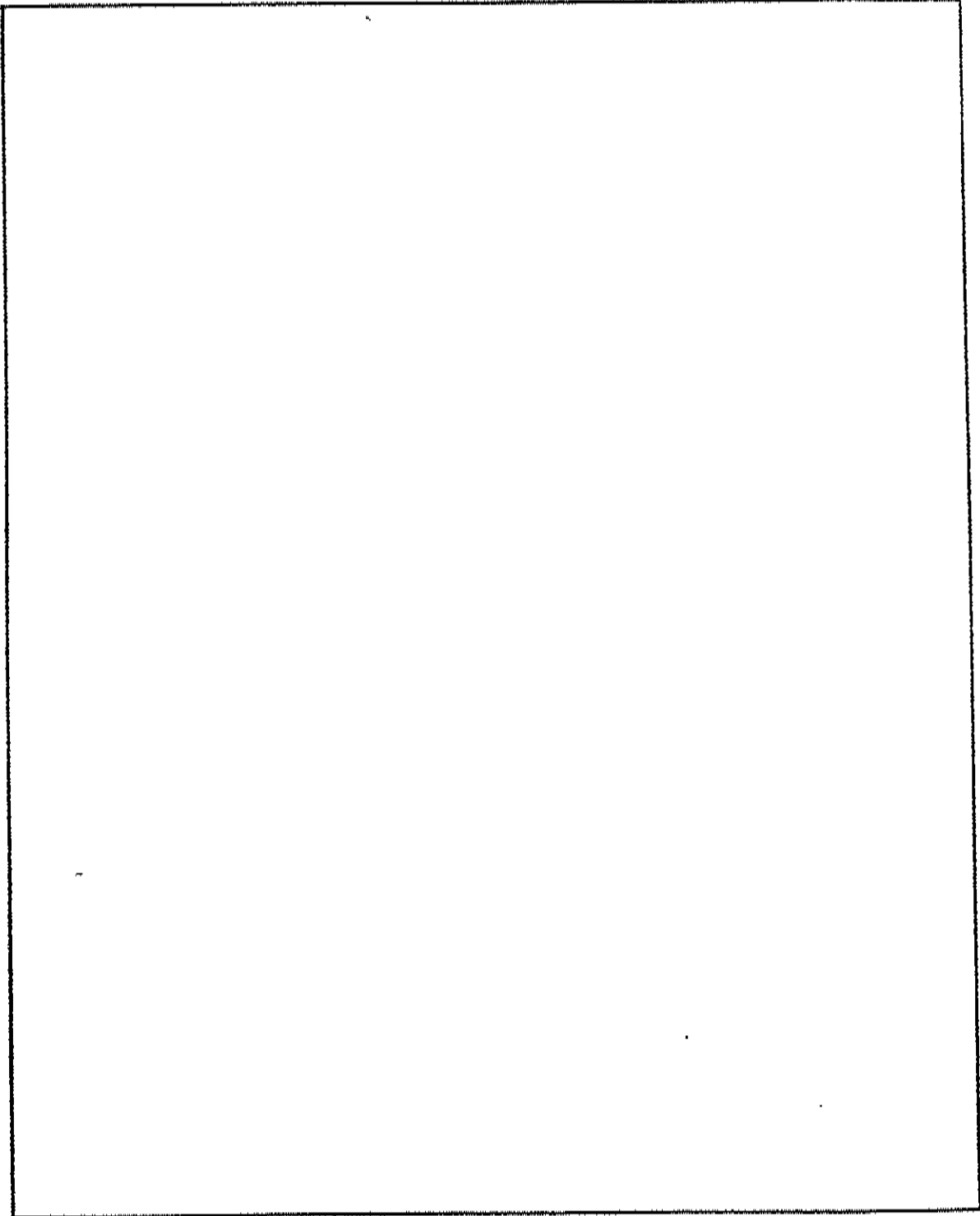
Instruction	Format	Semantics		
SUB A	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="text-align: center;">0 0</td><td style="text-align: center;">A</td></tr></table>	0 0	A	$AC \leftarrow AC - M[A]$
0 0	A			
STORE A	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="text-align: center;">0 1</td><td style="text-align: center;">A</td></tr></table>	0 1	A	$M[A] \leftarrow AC$
0 1	A			
ADD A	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="text-align: center;">1 1</td><td style="text-align: center;">A</td></tr></table>	1 1	A	$AC \leftarrow AC + M[A]$
1 1	A			
BRN	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="text-align: center;">1 0</td><td style="text-align: center;">A</td></tr></table>	1 0	A	If $AC \geq 0$ then branch to address A.
1 0	A			

Note: Put all your answers only in the spaces provided.

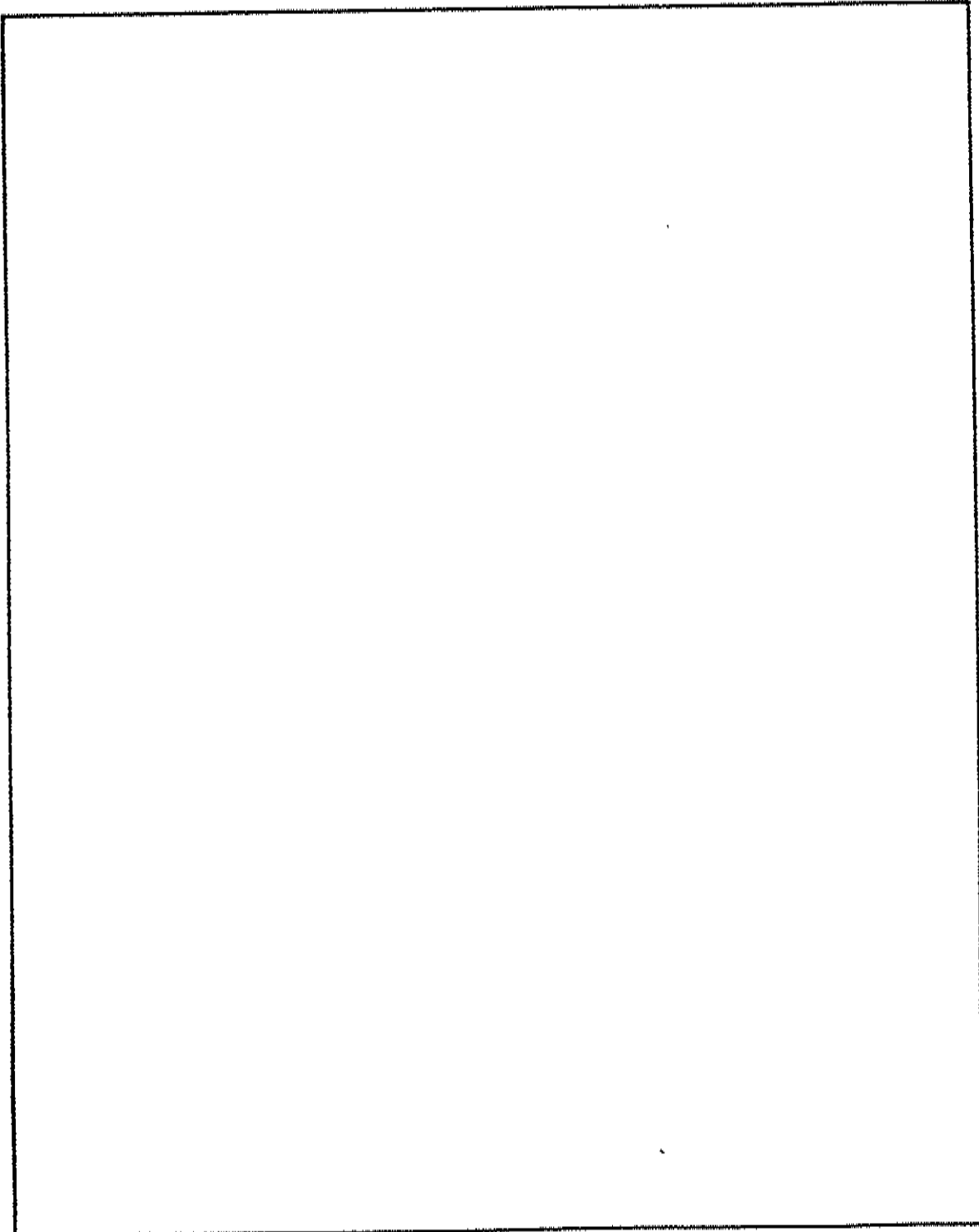
Be neat! You can work (scratch) on the back of the pages.

Problem #6

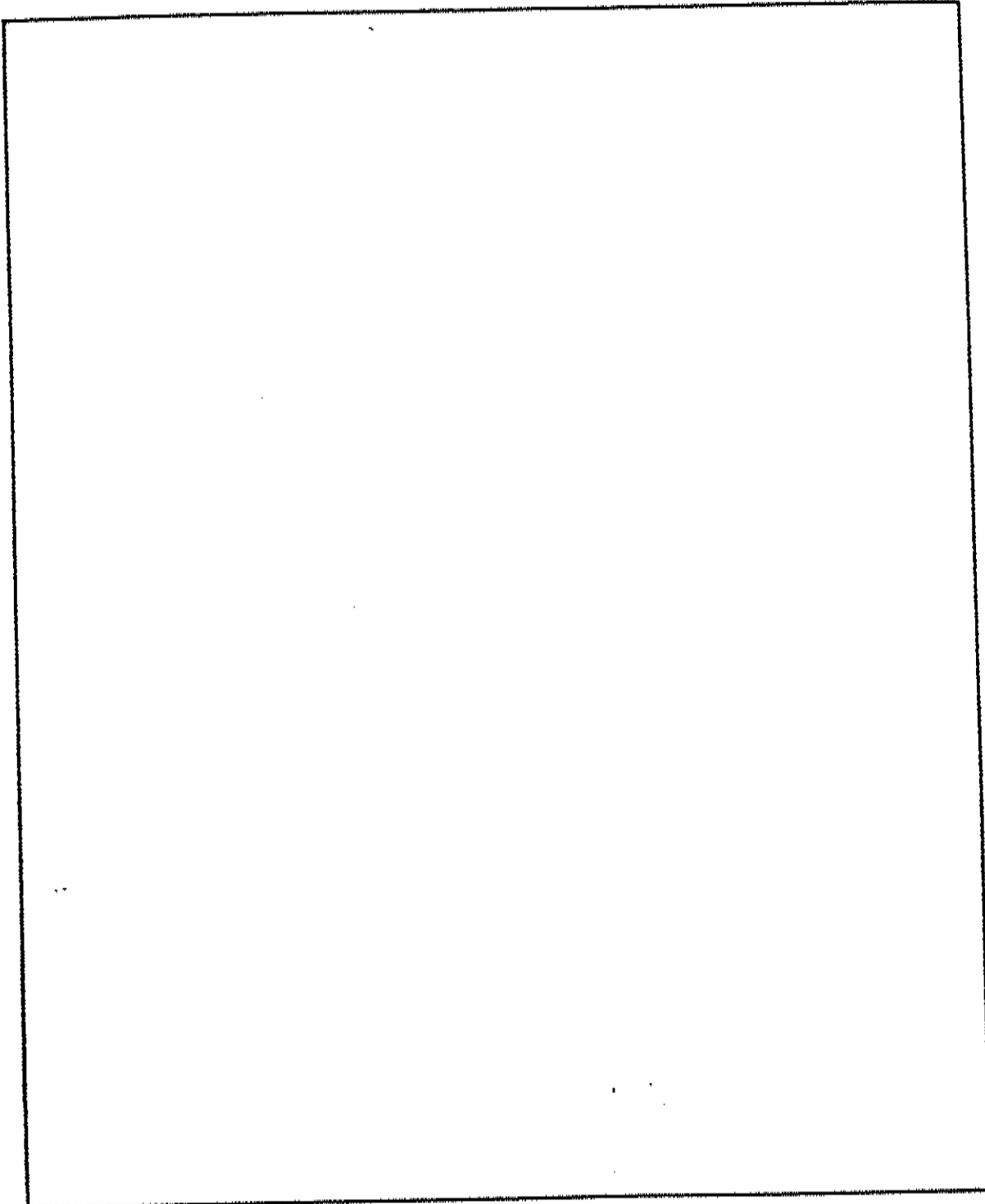
- A. Describe the basic components (flip-flops, etc.) of the computer in terms of NAND gates.



B. Draw a register diagram of the computer.

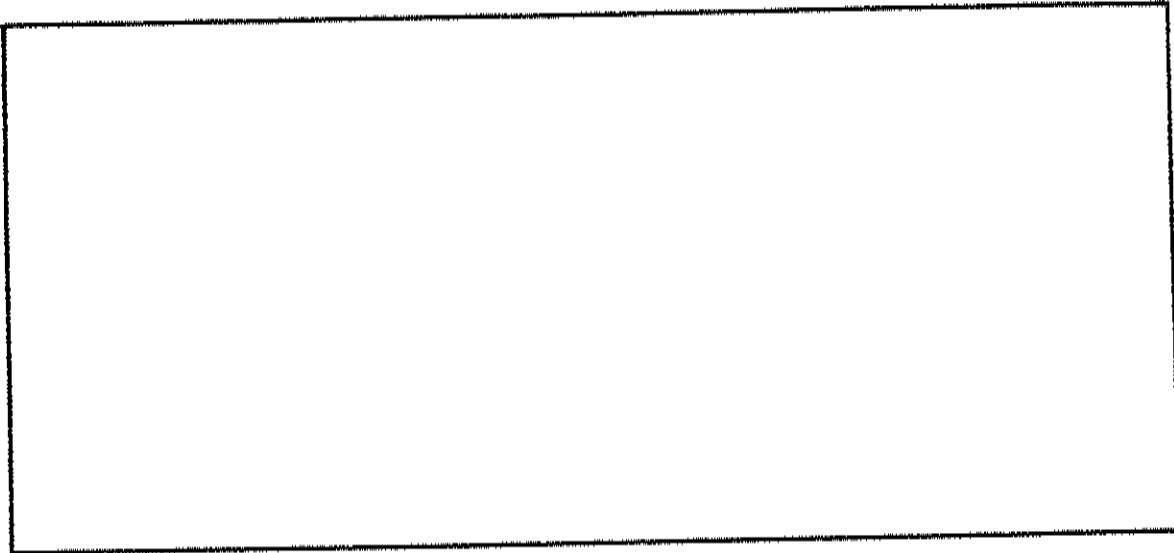


C. Draw a state diagram of the computer:

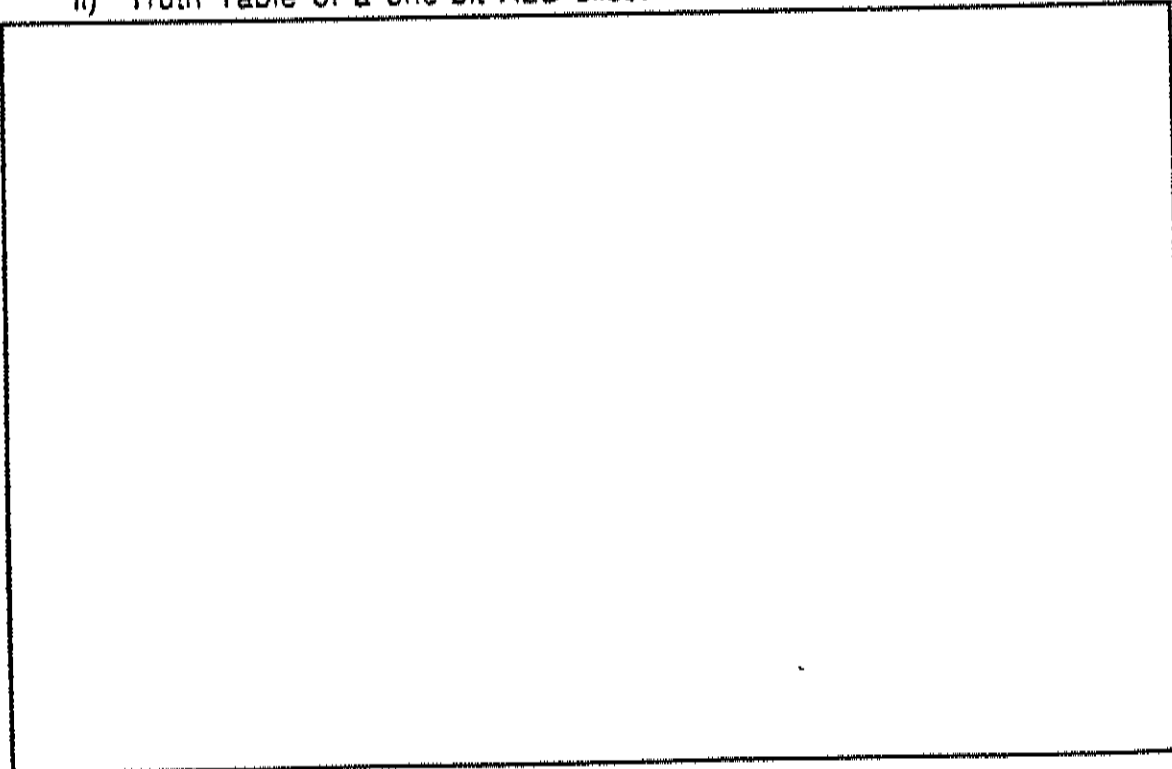


D. Draw a circuit of the ALU of the computer:

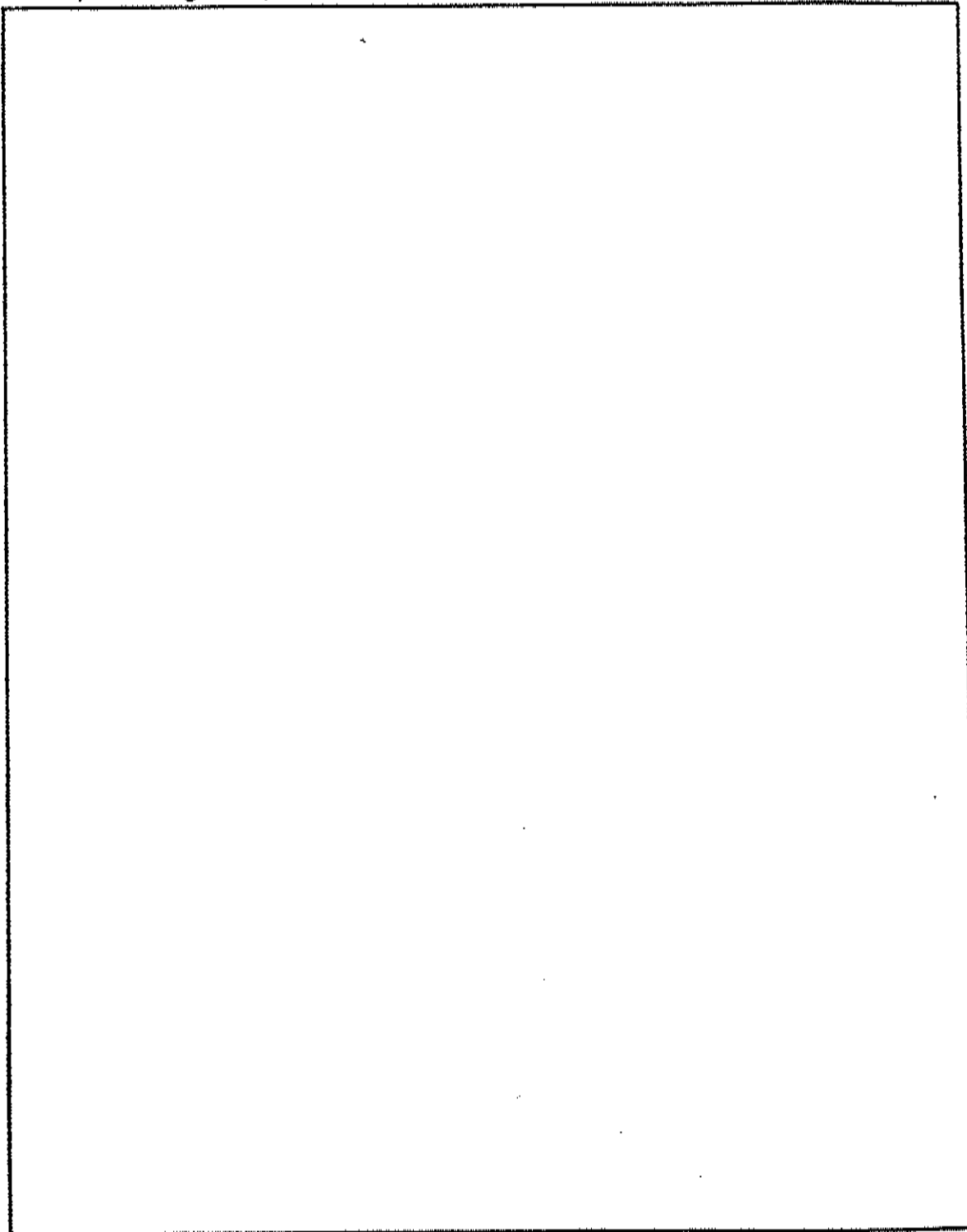
i) ALU in terms of "bit-slices":



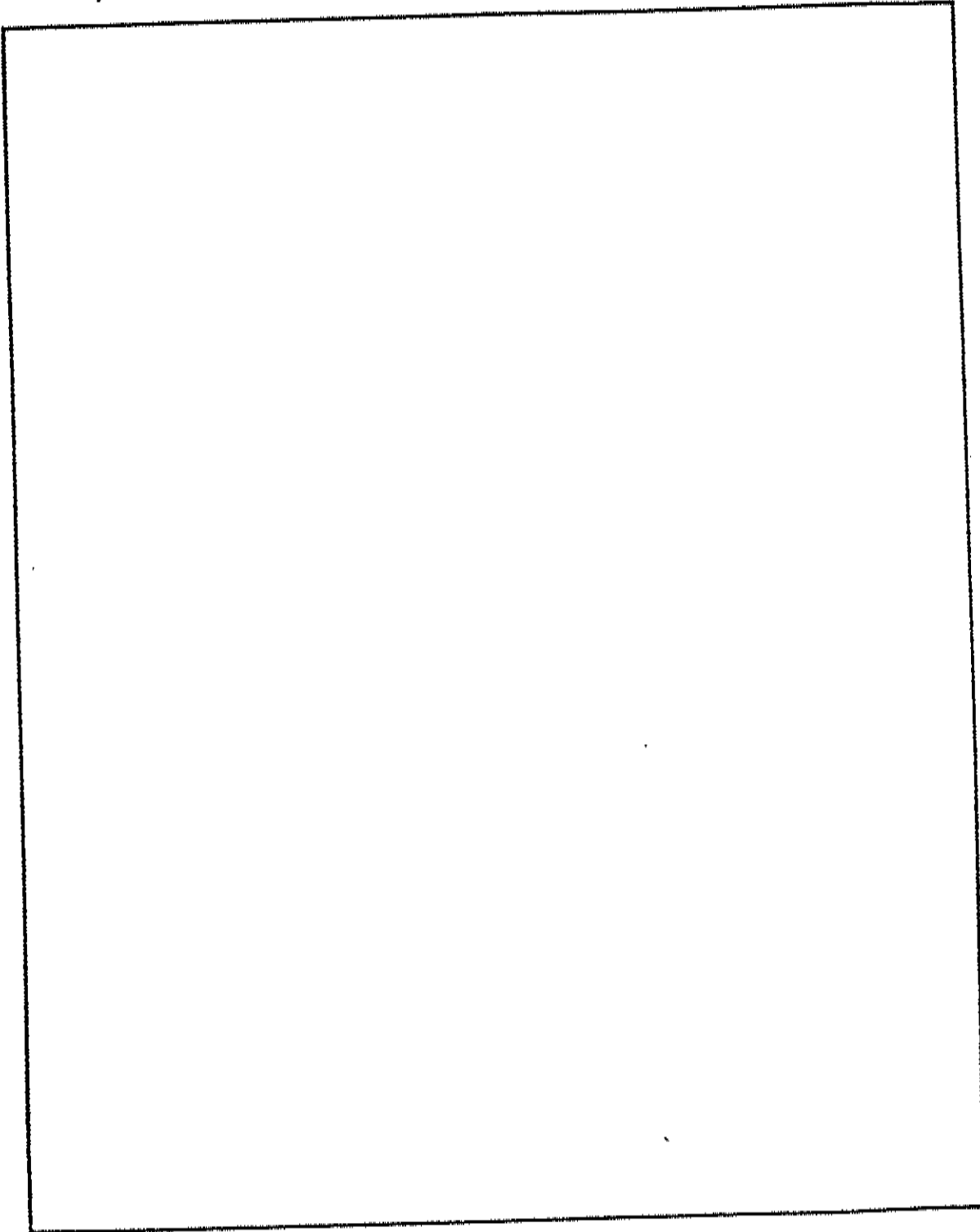
ii) Truth Table of a one-bit ALU slice:



iii) Karnaugh Map of a one-bit ALU Slice:

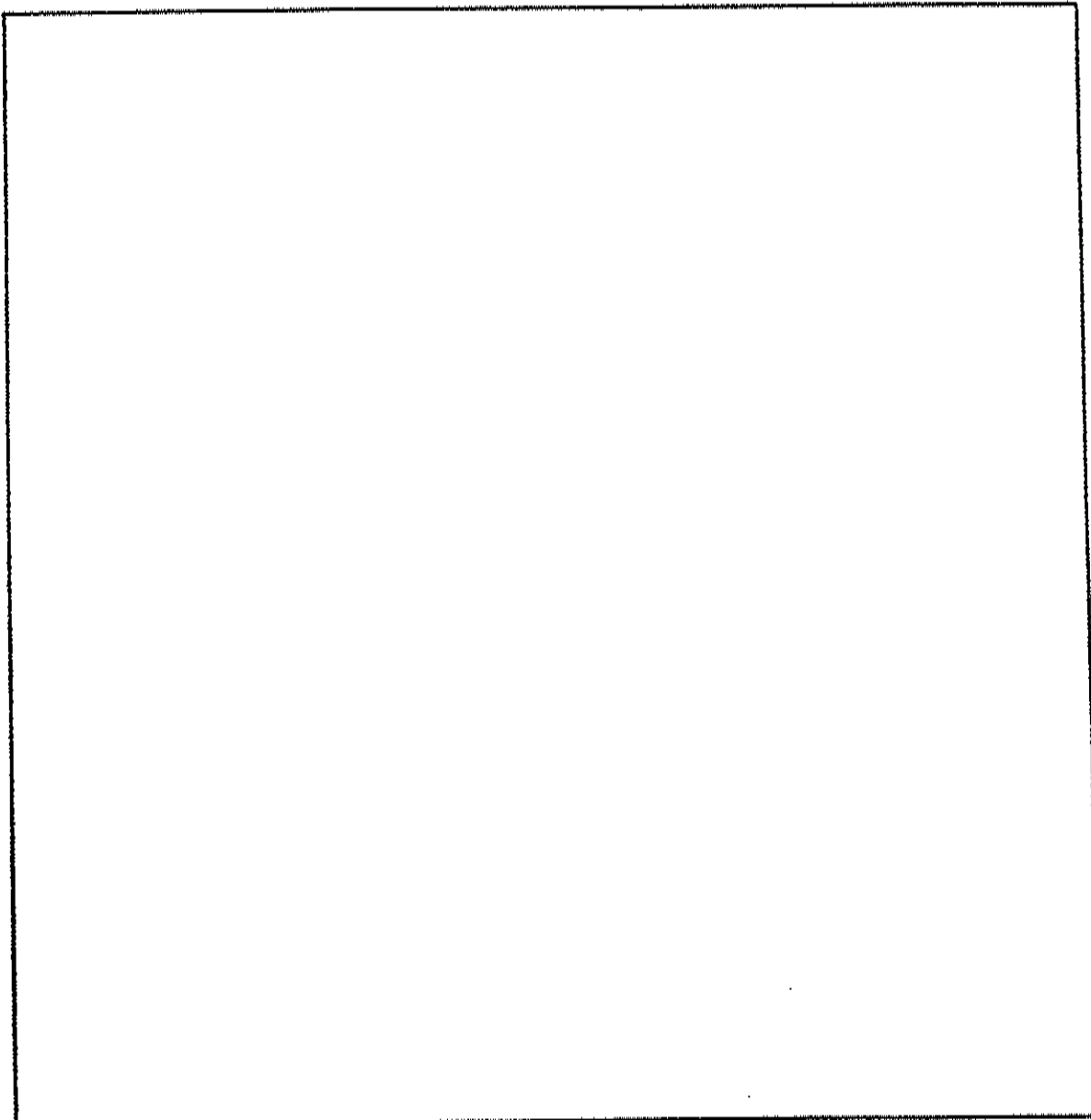


iv) NAND GATE (only) circuit of ALU Bit Slice:

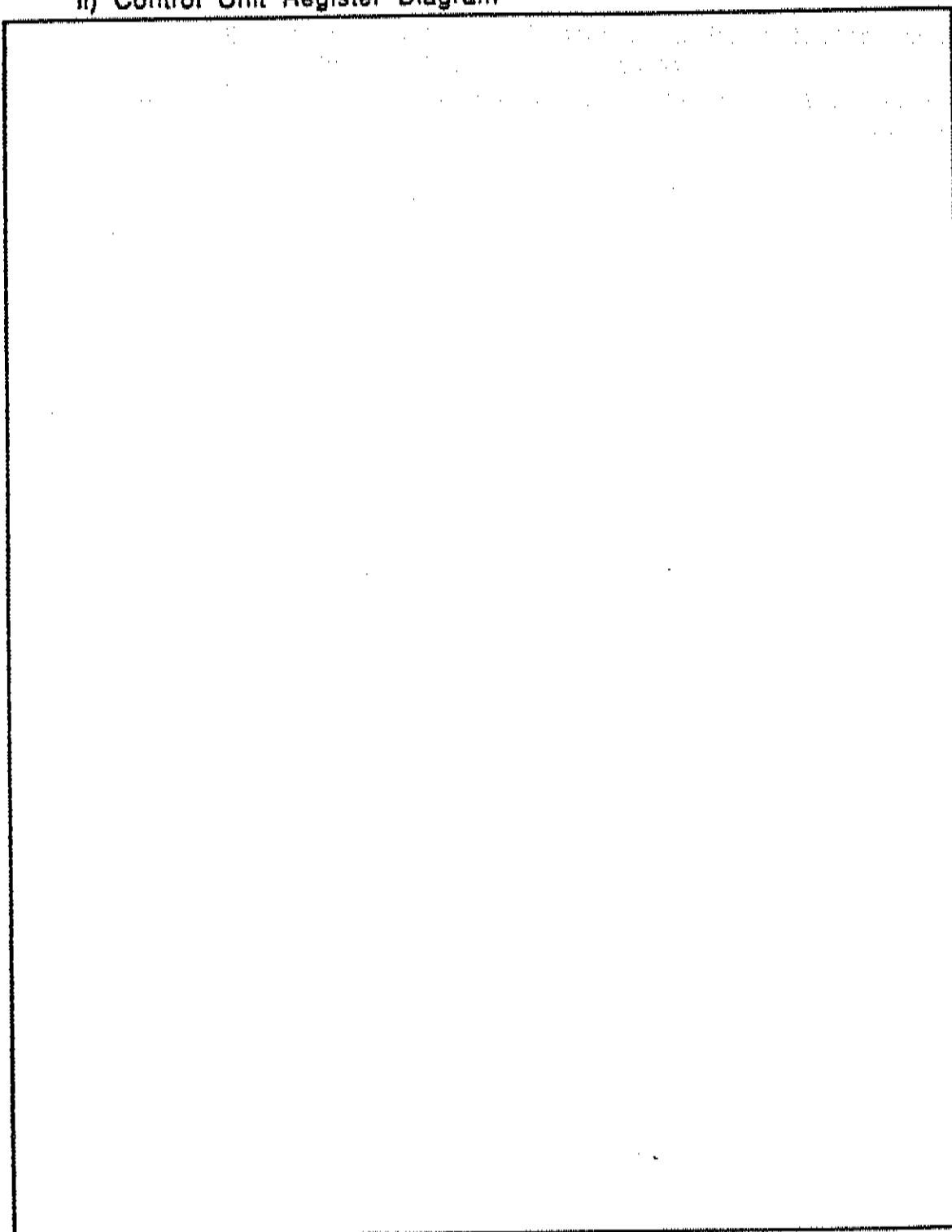


E. Given your state diagram and register diagram, show a complete circuit diagram of your control unit. You must identify and design all aspects of the control including the identification of the control register and its implementation in NAND gates. Carefully organize your work and employ and show all truth tables, K-maps, etc. as needed. Be organized and neat!

i) Label all the inputs and output to your control unit:



ii) Control Unit Register Diagram



20

Note: The notations x' and \bar{x} are equivalent.

a) State DeMorgan's Theorem for two variables A and B:

2

$(A \vee B)' = \bar{A} \wedge \bar{B}$ ✓, $A \wedge B = \overline{\bar{A} \vee \bar{B}}$ ✓

b) Prove a). Truth table

2

A	B	$A \vee B$	\bar{A}	\bar{B}	$\overline{A \vee B}$	$\bar{A} \wedge \bar{B}$	$A \wedge B$	$\overline{A \wedge B}$	$\bar{A} \vee \bar{B}$
0	0	0	1	1	1	1	0	1	1
0	1	1	1	0	0	0	0	1	1
1	0	1	0	1	0	0	0	1	1
1	1	1	0	0	0	0	1	0	0

c) Simplify the following using algebraic operations:
 Show your steps.

i) $(A + B)(A + C)$

2

$$\begin{aligned}
 &AA + AB + AC + BC \\
 &A + AB + AC + BC \\
 &A(1 + B + C) + BC \\
 &A + BC
 \end{aligned}$$

answer

$A + BC$

ii) $(AB' + A'B + A'B')$

$$\overline{A\bar{B} + \bar{A}B + \bar{A}\bar{B}}$$

answer

AB

2

$$\begin{aligned}
 &= (\overline{A\bar{B}})(\overline{\bar{A}B})(\overline{\bar{A}\bar{B}}) \\
 &= (\bar{A} + B)(A + \bar{B})(A + B) = (A\bar{A} + AB + \bar{A}\bar{B} + B\bar{B})(A + B) \\
 &= (AB + \bar{A}\bar{B})(A + B) \\
 &= (AAB + ABB + A\bar{A}\bar{B} + \bar{A}\bar{B}B) = AB + AB + 0 + 0 = AB
 \end{aligned}$$

d) Use a Karnaugh map to simplify the following:

$$ABCD + A'B'C'D' + A'BC'D' + A'B'C'D + A'BC'D + ABCD' + AB'CD' + AB'CD$$

2

		AB			
		00	01	11	10
CD	00	1	1	0	0
	01	1	1	0	0
	11	0	0	1	1
	10	0	0	1	1

$$AC + \bar{A}\bar{C}$$

answer

$$AC + \bar{A}\bar{C}$$

ABCD	0000
	0001
	0011
	0010



e) Given the following truth table with d representing the "don't care" condition, write a simplified function using a Karnaugh map:

2

A	B	C	
0	0	0	d
0	0	1	d
0	1	0	d
0	1	1	d
1	0	0	1
1	0	1	d
1	1	0	0
1	1	1	0

		AB			
		00	01	11	10
C	0	x	x	0	1
	1	x	x	0	x

$$\bar{B}$$



answer

$$\bar{B}$$

Problem #1

f) Given the following truth table:

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Write F in:

i) sum of products form:

2

ii) product of sums form:

2

	AB			
	00	01	11	10
0	0	1	1	1
1	1	0	0	1

$$\bar{C}B + A\bar{B} + \bar{B}C$$

~~(A+B+C)(\bar{B}+\bar{C})~~
 $(A+B+C)(\bar{B}+\bar{C})$



answer

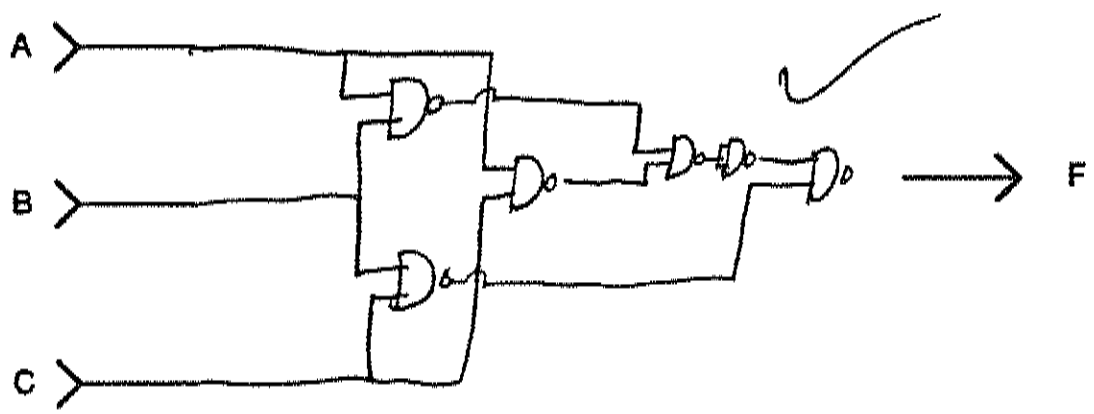
$$\bar{B}\bar{C} + A\bar{B} + \bar{B}C$$



answer

$$(A+B+C)(\bar{B}+\bar{C})$$

g) Draw a circuit using only 2-input NAND gates to implement $AB + BC + AC$:

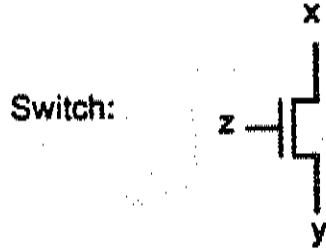
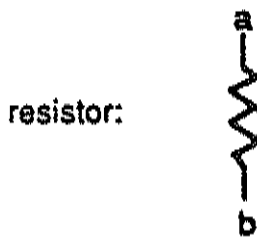


4

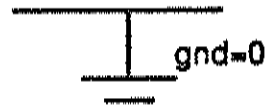
Problem #2 (15 points)

12

a) You are given two circuit elements, a resistor and a switch, and two power rails, VDD and GND called 1 and 0, respectively.



power rails



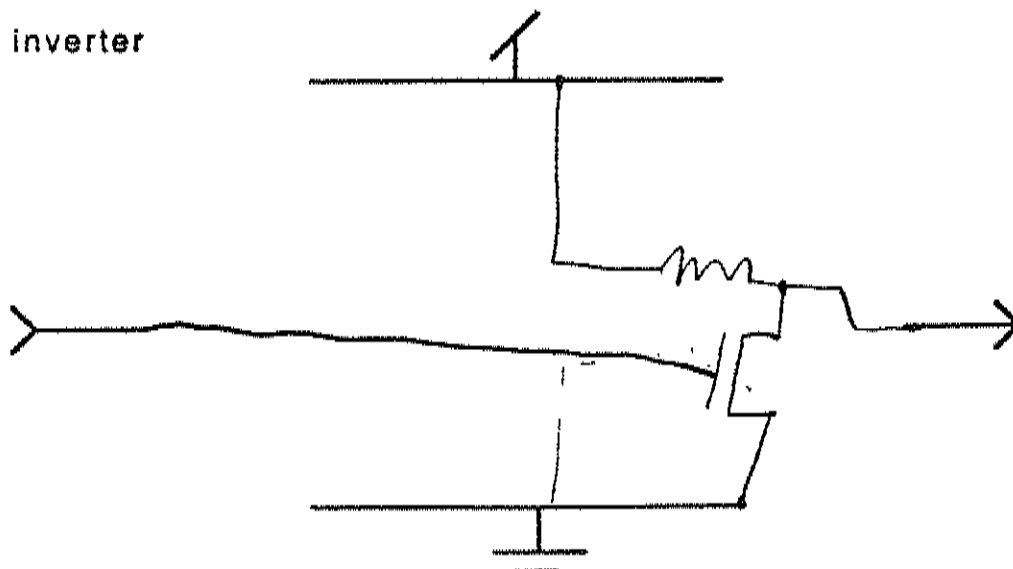
The circuit elements function as follows:

Switch: if terminal z is connected to 1 then x is connected to y.

Resistor: if terminal a is connected to 1 (or 0) then terminal b is forced to 1 (or 0). The switch is stronger than the resistor & wins in any conflicts.

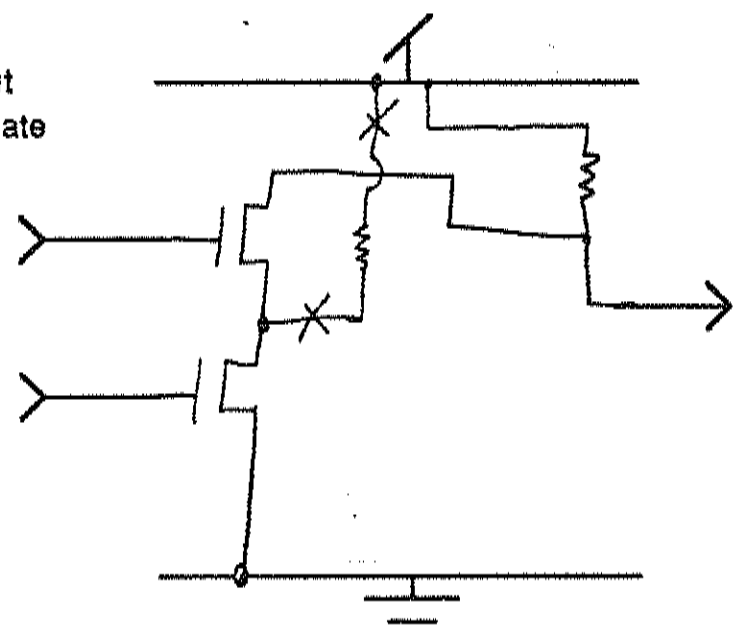
Show a circuit for each of the following. You may use as many elements as needed, but try to minimize the number of elements in each circuit.

inverter



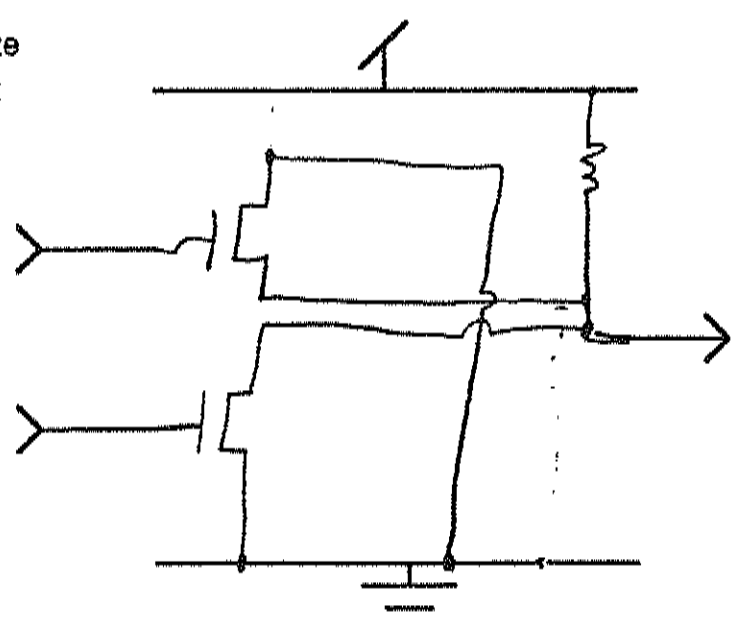
(2)

2- input
NAND gate



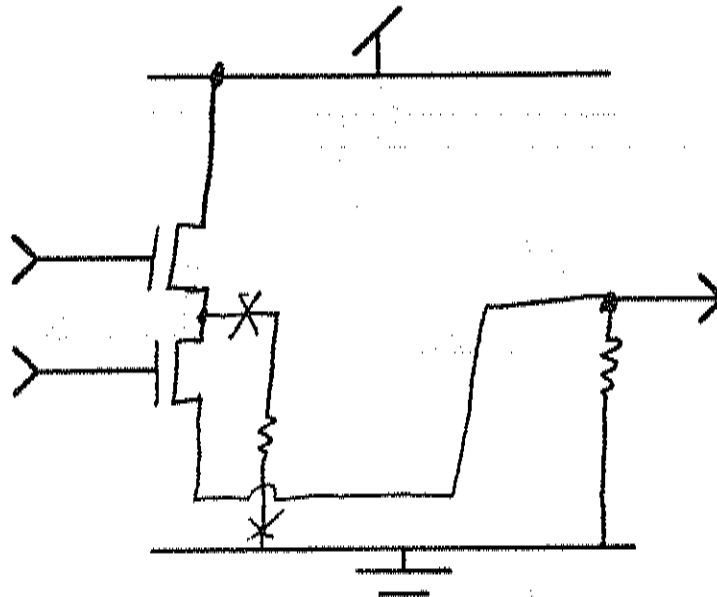
2

NOR gate
2-input

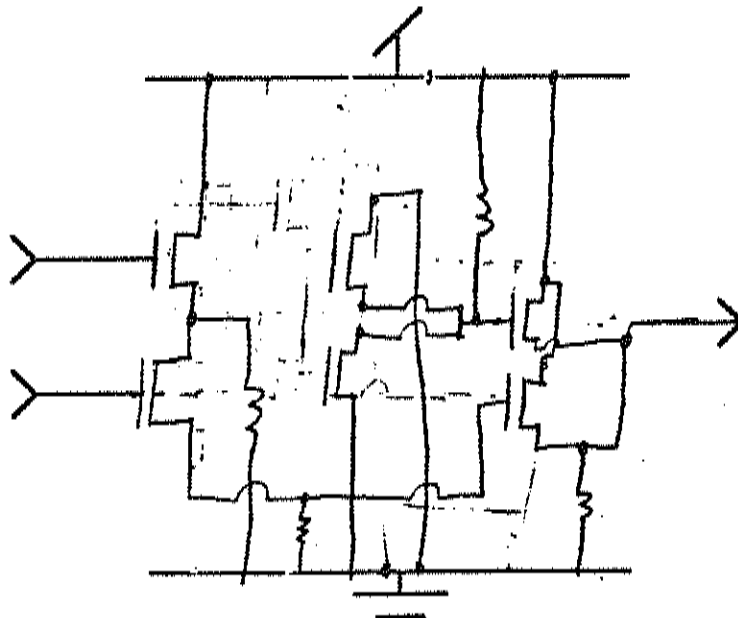


1

2-input
AND gate

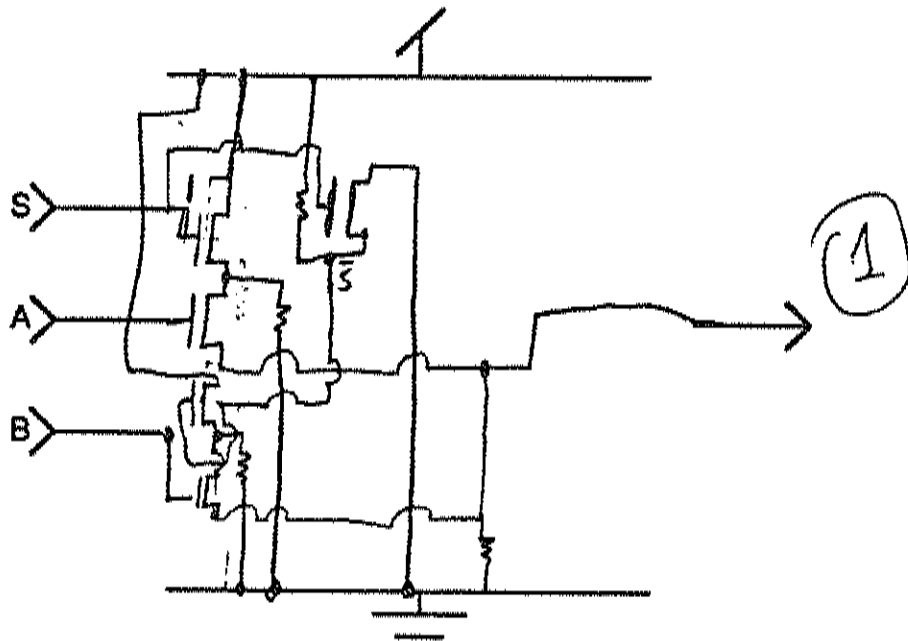


2-input
exclusive-NOR
gate

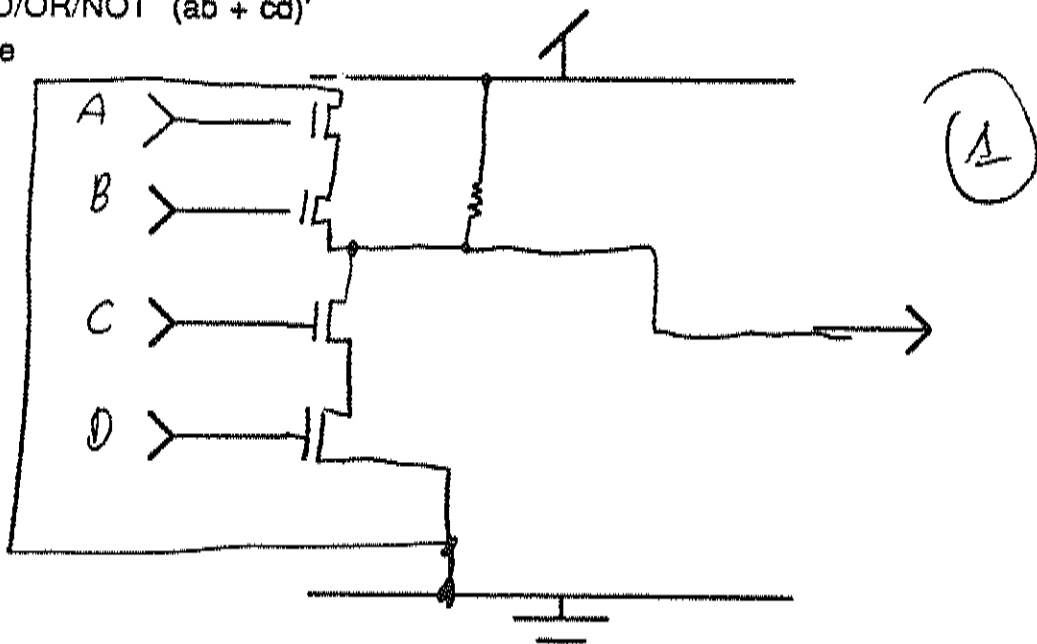


Problem #2

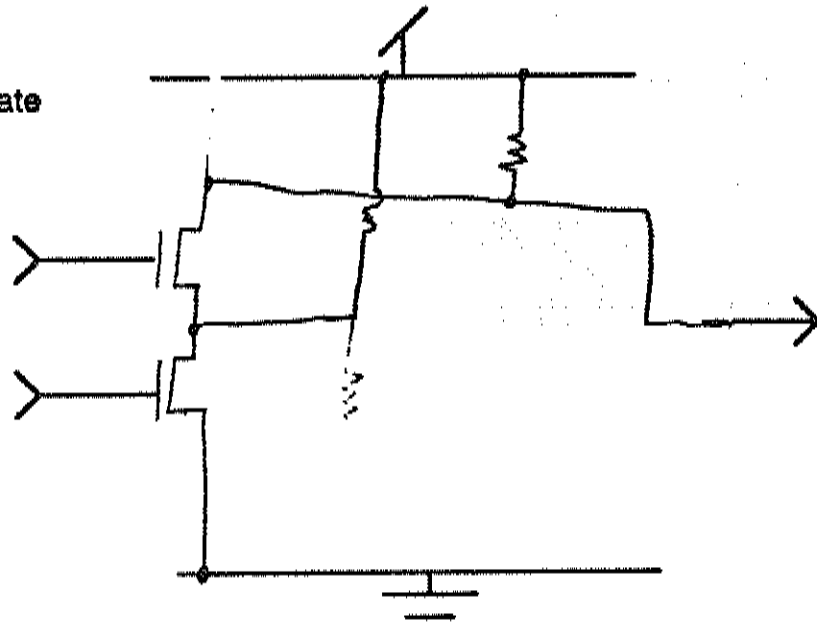
2 x 1
multiplexer



AND/OR/NOT $(ab + cd)'$
gate

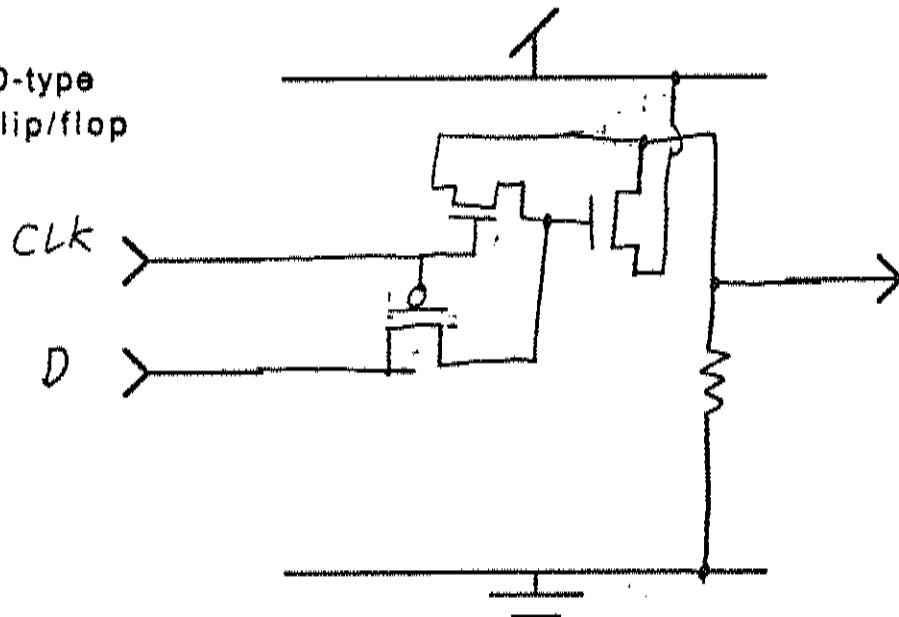


2-input
NAND gate



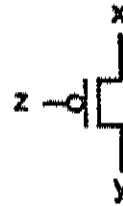
(1)

D-type
flip/flop



(1)

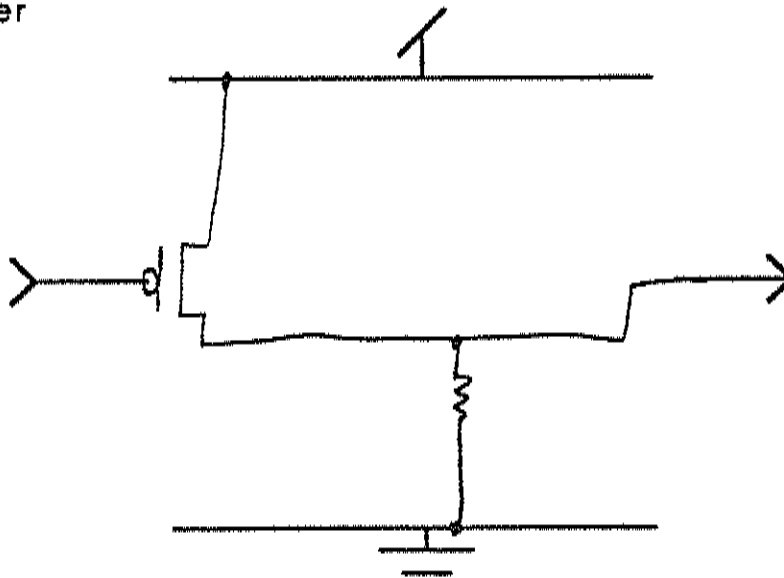
b) Now assume that instead of a resistor, you are given another switch with the following function:



if terminal Z is connected to 0 then X is connected to Y

Show a minimal circuit for each of the following. Again, try to minimize the number of elements in each circuit.:

Inverter

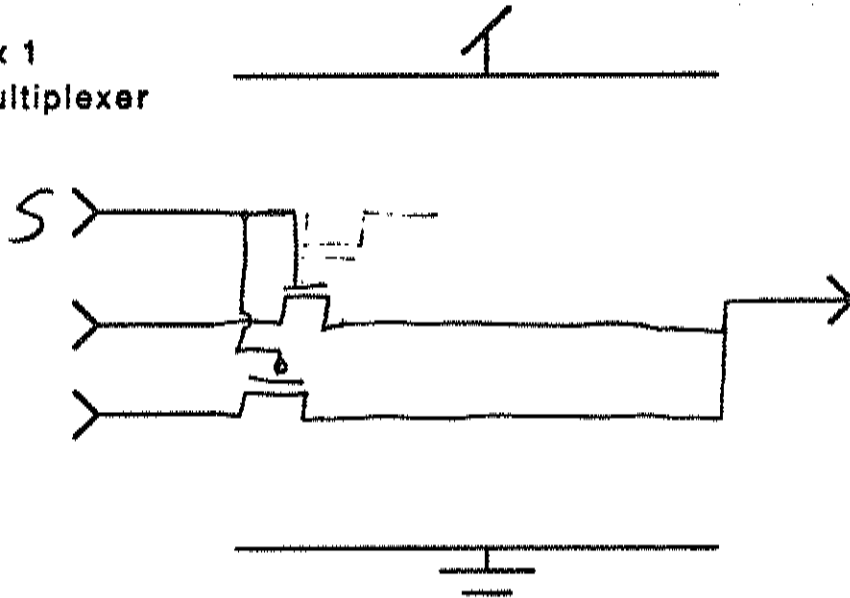


(2)

YOU SHOULD HAVE NOT USED RESISTORS

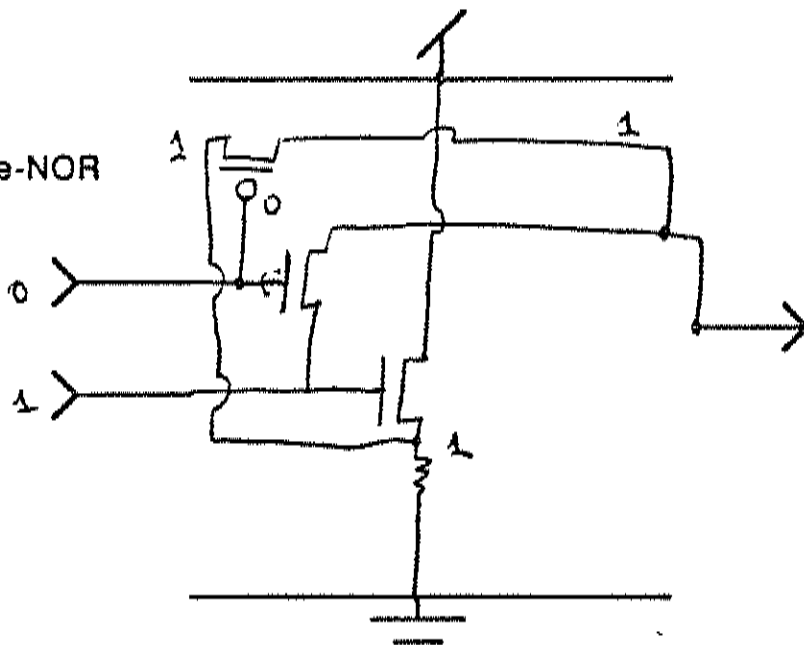
~~1~~ - 1
 IT SEEMS THE EXAM PAPER DOES NOT HAVE CORRECT SEQUENCE. (PG BEFORE PS).
 SO. INSTEAD OF -3 YOU GOT -1.

2 x 1
multiplexer



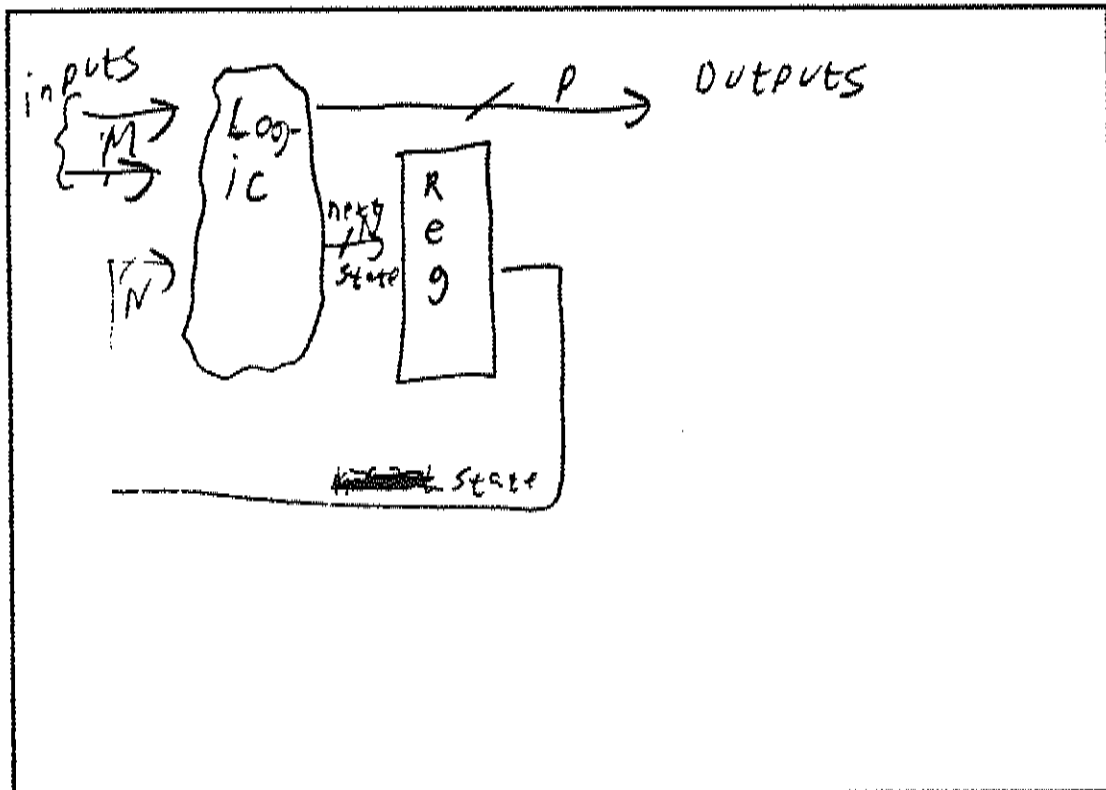
1

2-input
exclusive-NOR
gate



Problem #3 (40 points) 37

A. Show a register-level diagram of a Mealy FSM (Finite State Machine).



5

B. When should a Mealy machine be used?

When you want to minimize states needed to accomplish your goal; inputs need to be fairly stable and/or outputs need to be fairly insensitive to glitches. } X

3

C. Contrast the relative advantages and disadvantages of a Mealy vs. a Moore machine.

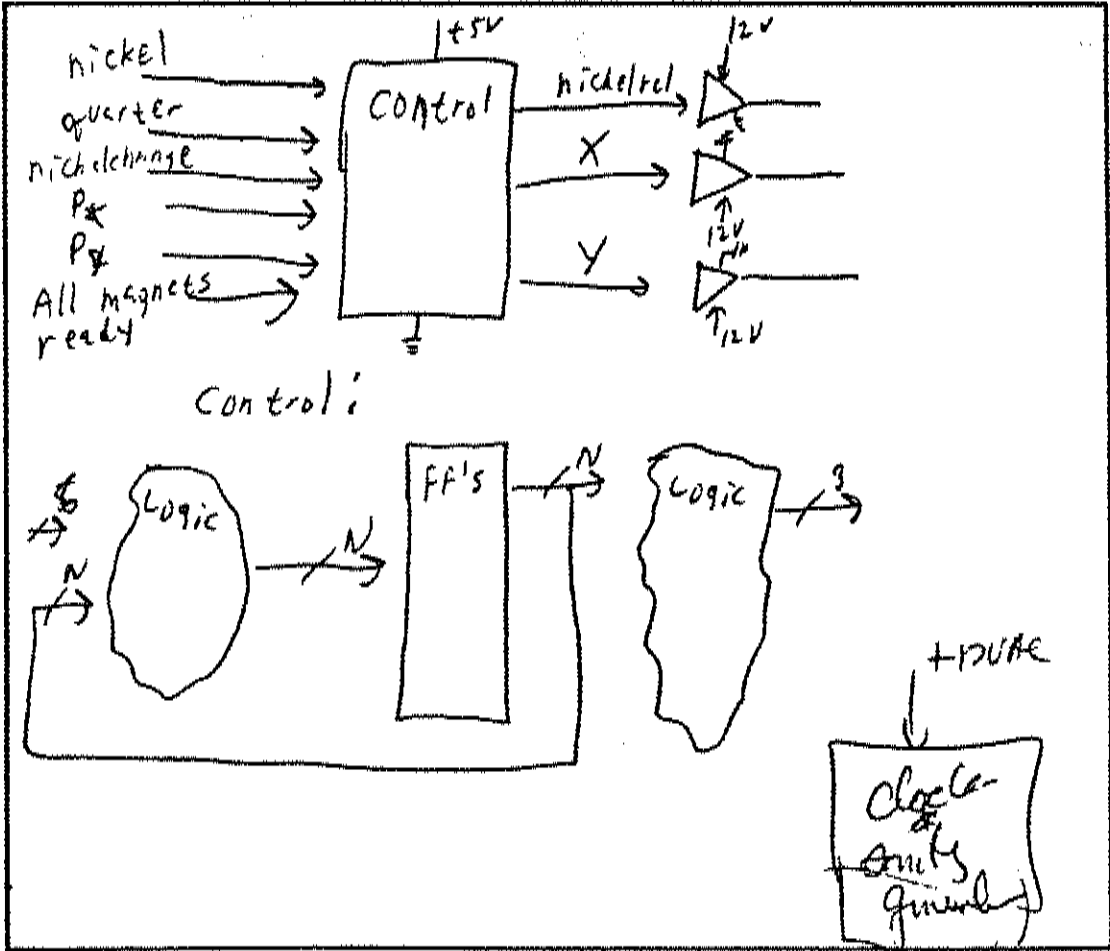
MEALY MACHINE

ADVANTAGES ↙	DISADVANTAGES ↘
Less states needed —	Outputs are more stable in a Moore machine —
Logic may be faster (less logic needed) —	Also Moore allows glitches on inputs —
more output combinations —	Simpler design for Moore —
Perhaps we <u>want</u> the outputs to change based on the inputs (i.e. have them change faster than register clock rate) —	more reliable for Moore —

D. A vending machine accepts nickels and quarters and dispenses two products X & Y and change in nickels. X costs 15¢ and Y costs 10¢.

There are five switches. Two sense the passage of quarters and nickels to the coin box. Two are push buttons the customer uses to select the desired product. One senses the passage of each nickel dispensed as change. There are three magnets. One to release nickels for change and two to release products X & Y. There are two power supplies: +5v.dc and a 12 volts AC at 60Hz. Design a complete circuit (controller) to control the vending machine. You may employ diodes, master-slave flip-flops, power operational amplifiers, gates and PLA's. You are required to use a Moore FSM — for this controller

i) Show a block diagram of your machine



clock
 Smt's
 Generator
 ↓
 CLK Smt's

000 → 001
011 → 100

Your number: 33

+ 5

010 → 011

00 → 01

01 → 10

10 → 11

iv) Show the relevant Boolean equations for your controller.

$$X^+ = P_x \bar{Y}$$

$$y^+ = P_y \bar{X}$$

Rn = nickel released (input)
Ry = nickel released (input)
Nr = release nickel (output)

$$C_2 = C_2 (R_n R_y)$$

$$\begin{cases} C_1 = C_1 (R_n R_y) & R_n \bar{C}_2 C_1 C_0 + R_n C_2 \bar{C}_1 + R_n C_2 \bar{C}_0 + R_y C_2 \\ C_0 = C_0 (R_n R_y) & R_n \bar{C}_1 C_0 + R_n C_1 \bar{C}_0 + R_y \bar{C}_1 C_0 + R_y C_1 \bar{C}_0 \\ & R_n \bar{C}_0 + R_y \bar{C}_0 \end{cases}$$

quarter adds 4 to low bits, 1 to high bits

$$N_r = C_2 + Y \bar{C}_2 C_1 C_0$$

$$R_y = Y \bar{C}_2 C_1 \bar{C}_0$$

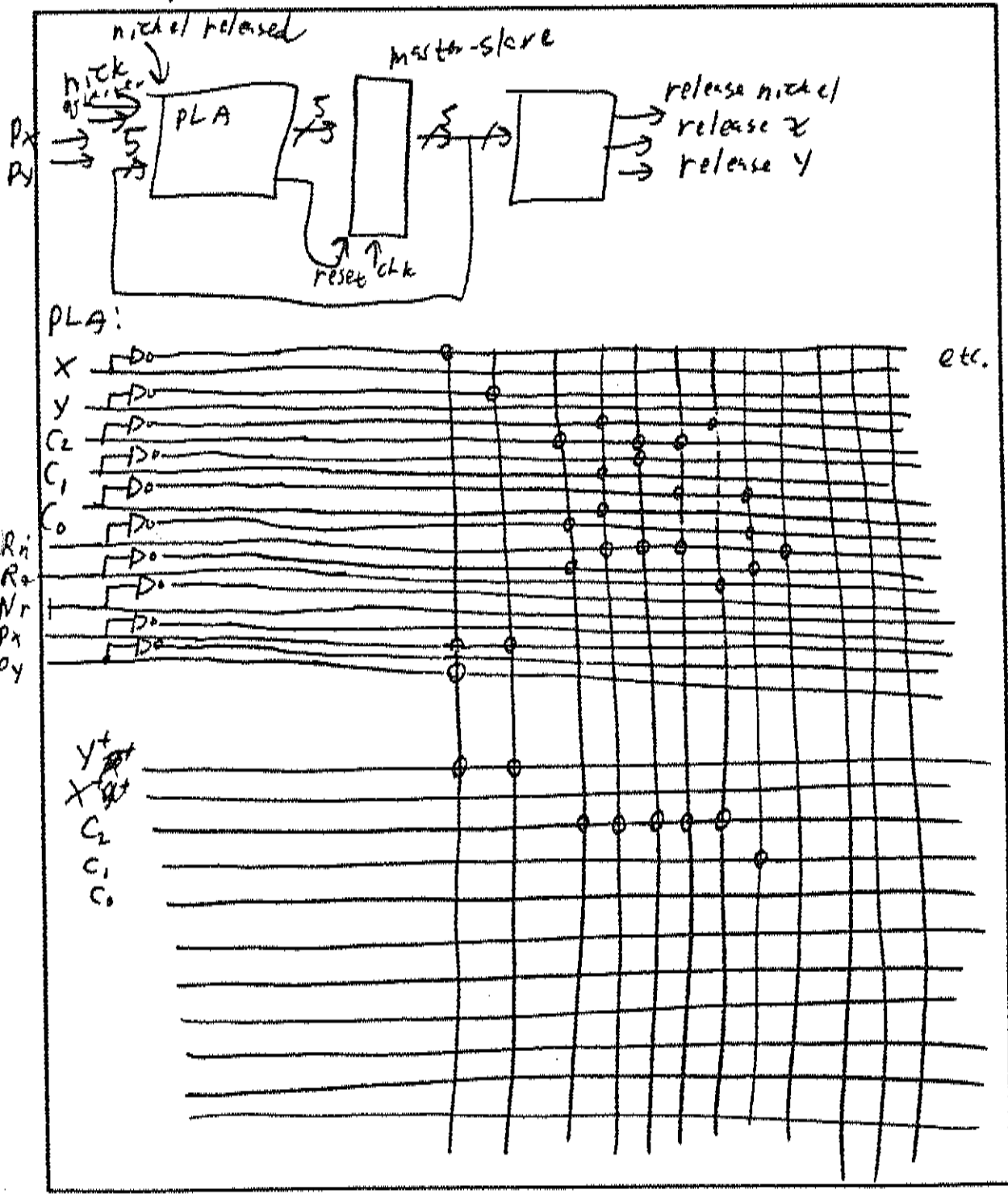
$$R_x = X \bar{C}_2 C_1 C_0$$

$$\text{Reset_dfa} = R_y + R_x$$

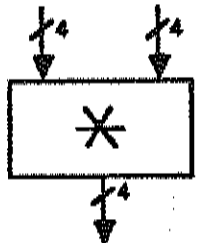
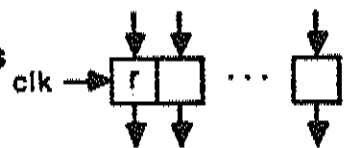
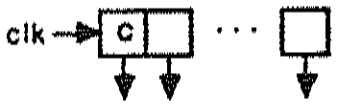
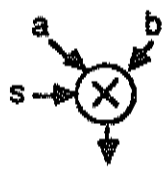

Problem #3

v) Show the circuit diagram of your controller. Be sure to use a PLA to implement the main FSM.

5

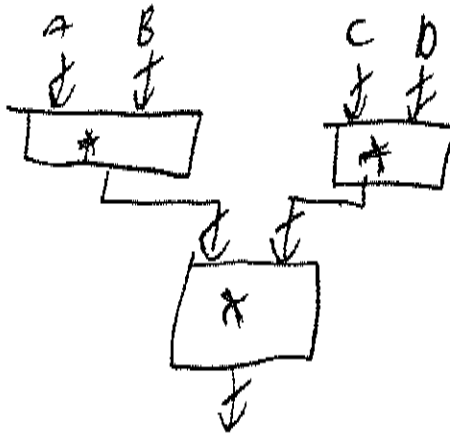


You are given four 4-bit numbers in read-only registers x_1 , x_2 , x_3 , and x_4 , along with a supply of parts. Each part has a price in dollars (\$) and a propagation delay in nanoseconds (ns). You may use as many of each part as you need. The four numbers are in unsigned fractional representation

Part	Symbol	Price	Delay	Notes
multiplier		6	10	All normalization & rounding is done within the multiplier.
read/write register bits		0.1 per bit	1	clk is a load signal. You do not need to show a circuit to drive clk.
counter bits		0.1 per bit	1 per bit	clk is a clock signal, as with the register you do not need to show a circuit to drive clk. The total delay for a counter is 1ns X # of bits.
2x1 Multiplexer		0.1	1	s is the select input, you do not need to show a circuit to generate s.
inverter		0.1	1	

You can assume that the register & counter bits start off initialized to all zeros.

- a) Draw a circuit that will multiply the four numbers in minimal time. The result need not be latched. What is the time if the circuit were extended to handle N 4-bit numbers? What is the price in each case? 12



for N, the
 answer would be
 $\lceil \log_2 N \rceil - 1$

Price for N:

$6(N-1)$

Price for four:

18

Time for four:

20

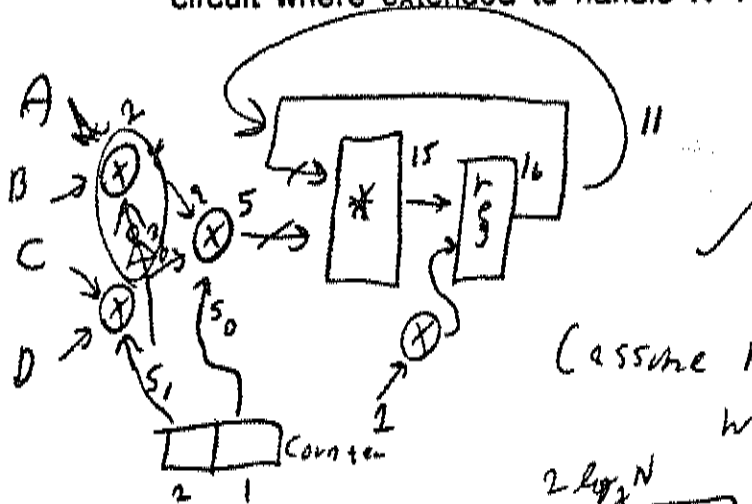
Time for N:

$10(\log_2 N)$

$4 \rightarrow 2$

(14)

b) Draw a circuit that will multiply the four numbers at a minimal price. Show the prices & times for the four numbers and if the circuit were extended to handle N 4-bit numbers.



(assume reg can be loaded w/ 1)

$$\begin{aligned}
 & 2 + \frac{2 \log_2 N}{3} + 0.1 + 0.1 + \frac{\log_2 N}{2} + 0.1 + 4 \times 0.1 \\
 & \quad + 6 \\
 & = 0.3 + 0.1 + 0.2 + 0.4 \\
 & = 0.4 + 0.2 + 0.4 \\
 & = 1.0 + 6
 \end{aligned}$$

$$\begin{array}{r}
 10 + 1 + 2 \\
 13 \\
 14 \\
 \frac{4}{56} \\
 \underline{\quad} \\
 + 2 \\
 11N +
 \end{array}$$

Price for four:

7

Price for N:

$0.4 + 6 + 3 \log_2 N$

Time for four:

64

Time for N:

11 +

X - 1

c) Now assume that each nanosecond of delay costs \$.05. What is the combined cost (cost due to price PLUS cost due to delay time) of your answer in a) & b) for four numbers?

combined cost of a):

combined cost of b):

Is there a circuit for four numbers that has less cost, under this measure?

Draw the circuit:

combined cost:

Problem #5 (20 points)

a) Convert the following from decimal to 8-bit two's complement form:

$$-75_{10} = \boxed{10110101} \checkmark \quad 2$$

b) Convert the following from two's complement to decimal:

$$11100001_2 = \boxed{\cancel{-15_{10}}}$$

c) Convert from decimal to two's complement fixed point:

$$0.65625_{10} = \boxed{0.10101} \checkmark \quad 2$$

neg 1 by 16

d) Multiply the following two's complement integers.
Show all partial products. Show an 8-bit result:

3

$$\begin{array}{r} 1011 \\ \times 1001 \\ \hline 000101 \\ 11011 \\ 00000 \\ 0101 \\ \hline 0100011 \end{array}$$

using Booth's algorithm.

answer

0100011 ✓

4

e) Consider two 4-bit two's complement fractional numbers in the following form: S. X X X (sign-bit, followed by the binary point, followed by three fraction bits), multiply the two numbers. Show all partial products. Express your result in the same form, i.e. S. X X X, using

i) truncation:

11.110 ✓

ii) rounding:

11.111 ✓

$$\begin{array}{r}
 0.010 \\
 X \underline{1.011} \\
 \hline
 1111110 \\
 0000000 \\
 000010 \\
 11110 \\
 \hline
 11.110110
 \end{array}$$

f) Consider two normalized floating-point number systems, NS1 and NS2.

4

NS1 has four bits of unsigned fractional mantissa in radix-2, and two bits of unsigned exponent.

NS2 has four bits of unsigned fractional mantissa organized as two radix-4 digits, and two bits of unsigned exponent.

OK

assuming no hidden bit.

Fill in the following table for each system:

	NS 1	NS 2
smallest mantissa	.1000 ✓	.0100 ✓
largest mantissa	1111 ✓	1111 ✓
largest exponent	11 ✓	11 ✓
largest representable value	15.0 ✗	240.0 ✗
number of fractions	$\frac{1 \times 2^4}{2} = 16$ ✗	16 ✗
number of exponents	✓ 4	4 ✓
number of values	✗ 64	64 ✗

iii) Karnaugh Map of a one-bit ALU Slice:

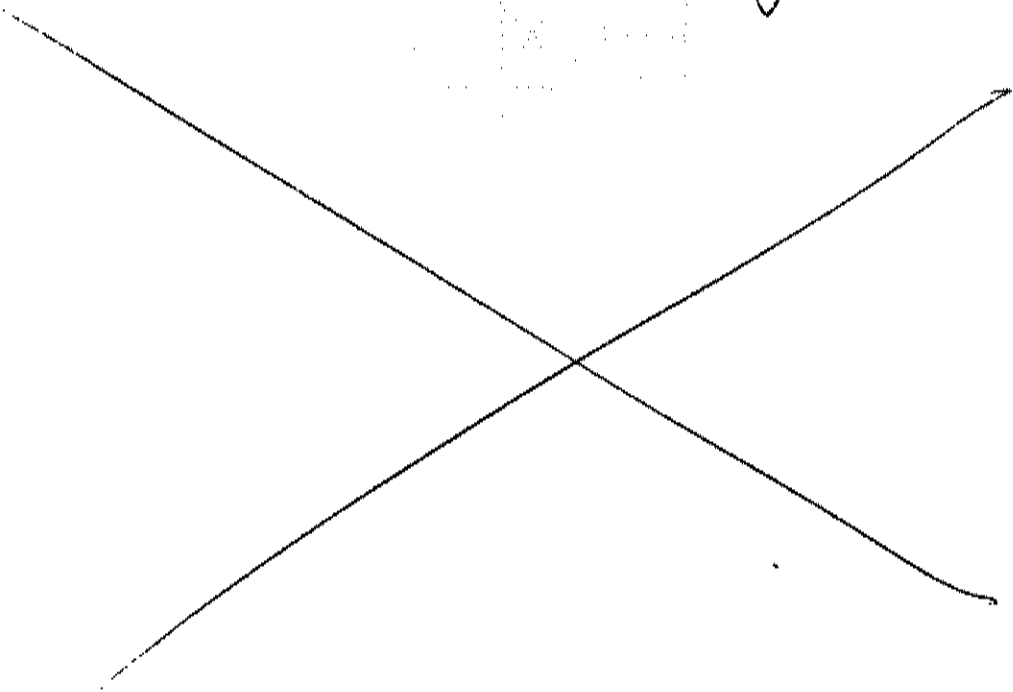
Karnaugh Map:

		S			
		00	01	11	10
c	0	0	1	0	01
	1	1	0	1	0

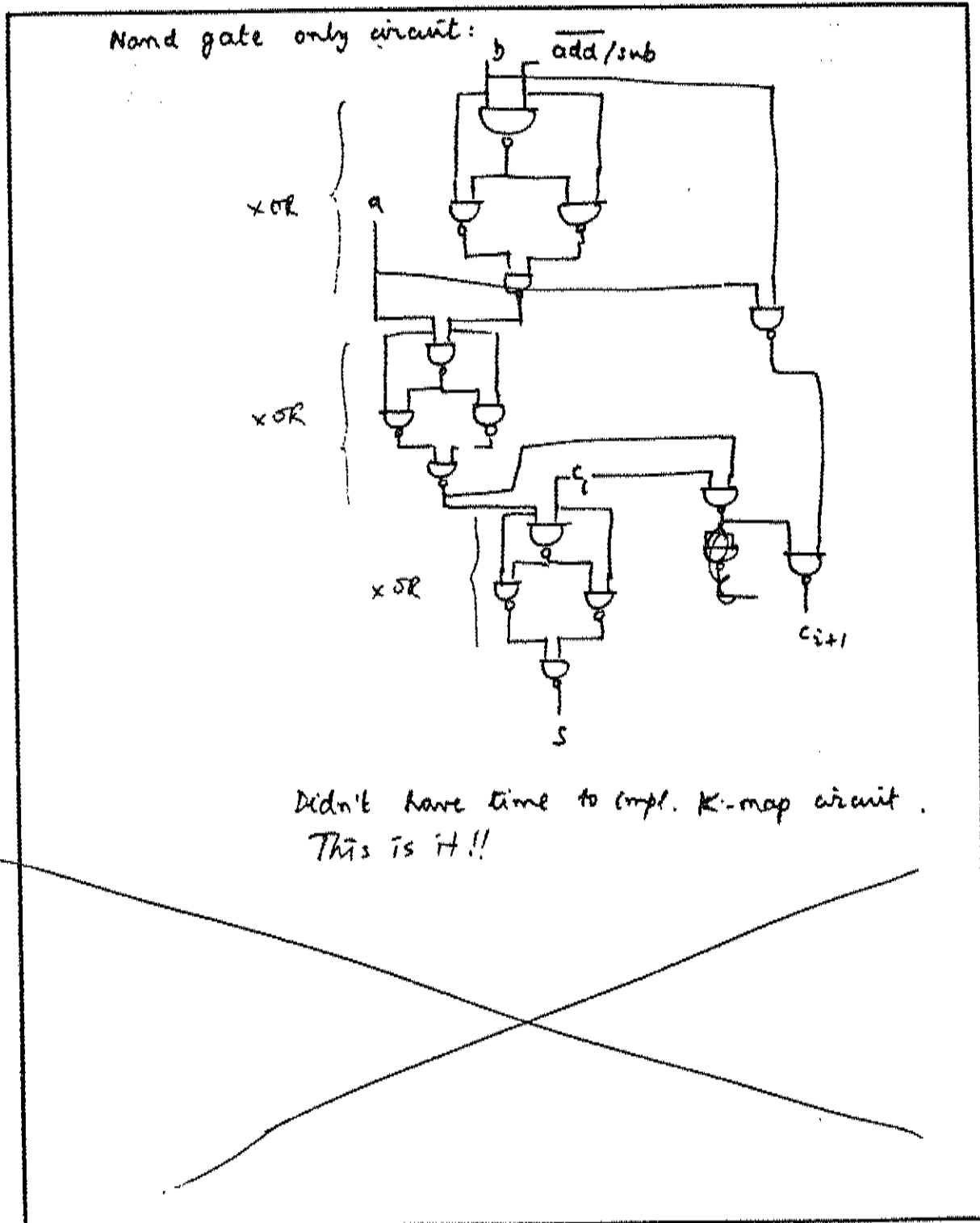
$S = a \oplus b \oplus c$

		c			
		00	01	11	10
c	0			1	
	1		1	1	1

$c = (a+b)c + ab$

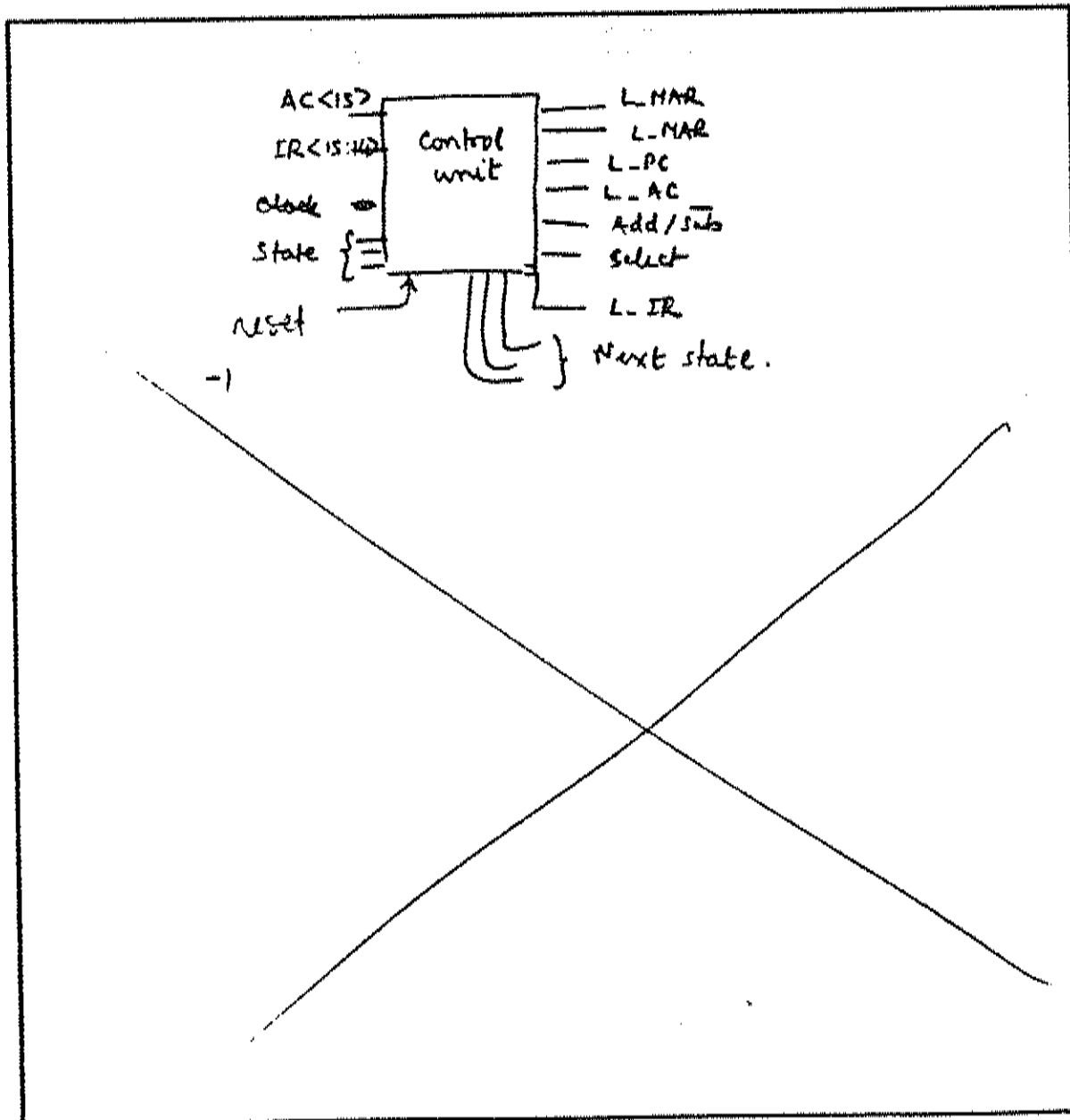


iv) NAND GATE (only) circuit of ALU Bit Slice:



E. Given your state diagram and register diagram, show a complete circuit diagram of your control unit. You must identify and design all aspects of the control including the identification of the control register and its implementation in NAND gates. Carefully organize your work and employ and show all truth tables, K-maps, etc. as needed. Be organized and neat!

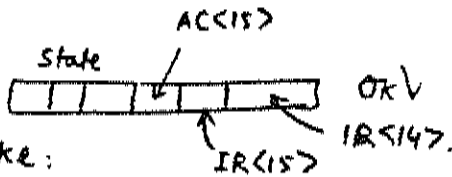
i) Label all the inputs and output to your control unit:



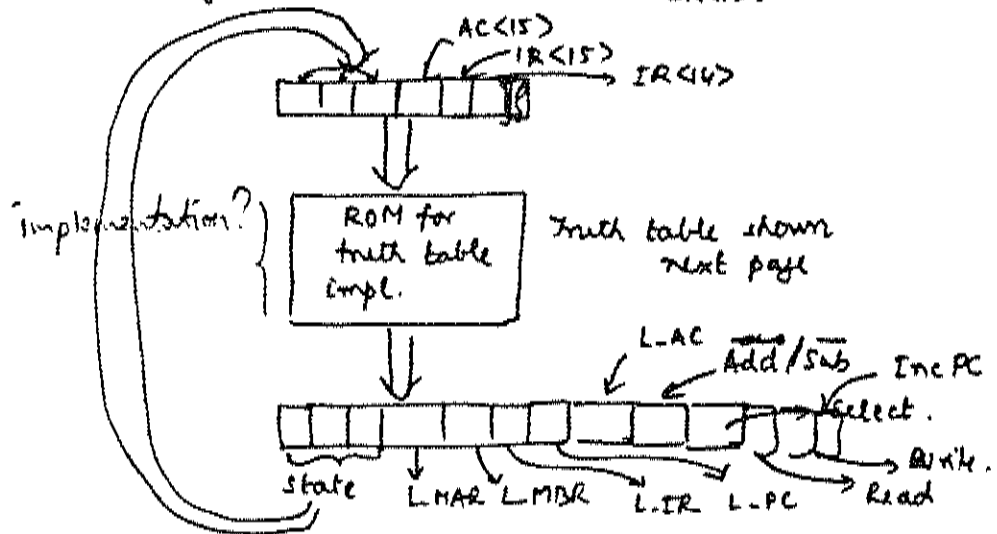
ii) Control Unit Register Diagram

Control unit register level diagram:

Control unit register looks like:



The block diagram looks like:



**HARDWARE CORE EXAM
SPRING 1989**

HARDWARE PRELIM EXAM

Spring 1989

Your code number:

General Instructions:

The exam lasts 3 hours = 180 minutes.

The exam has 12 pages.

Do all your work on these exam papers; use backsides if necessary.

Read the problem statements carefully, at least twice.

You should not need to ask questions.

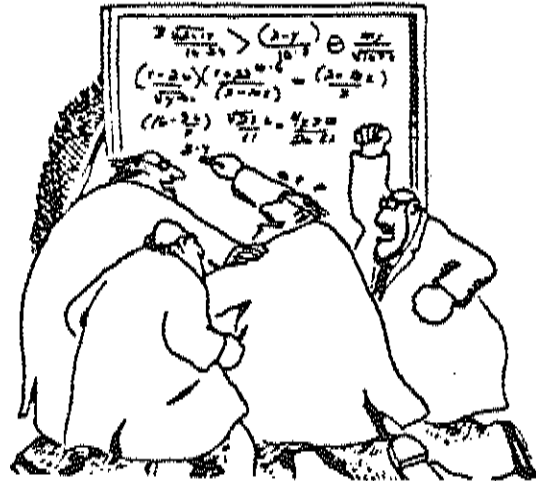
If something seems unclear, state your assumptions.

The indicated points give you an idea how long a problem should take (minutes).

There are 10 problems adding up to 200 points, so you have some choice.

Completing 180 points is considered a "perfect 10".

*Don't panic.
Good luck!
CHS*



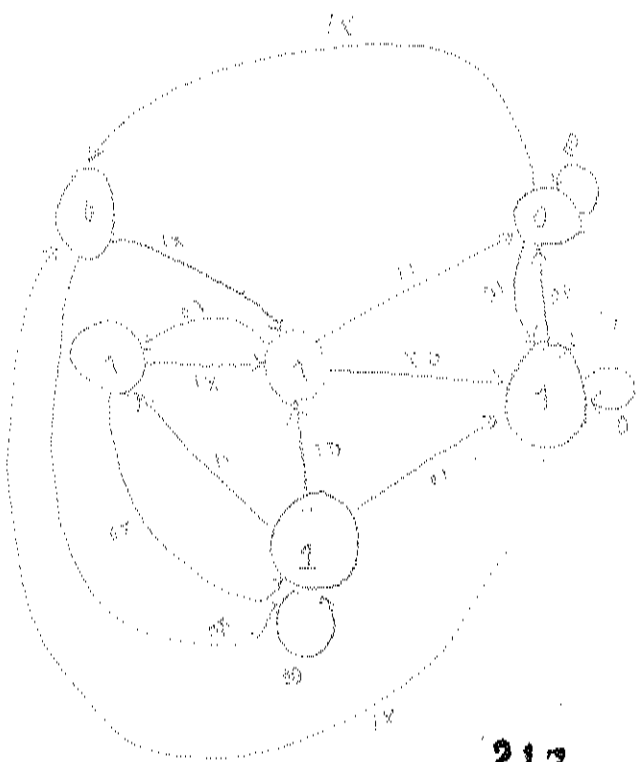
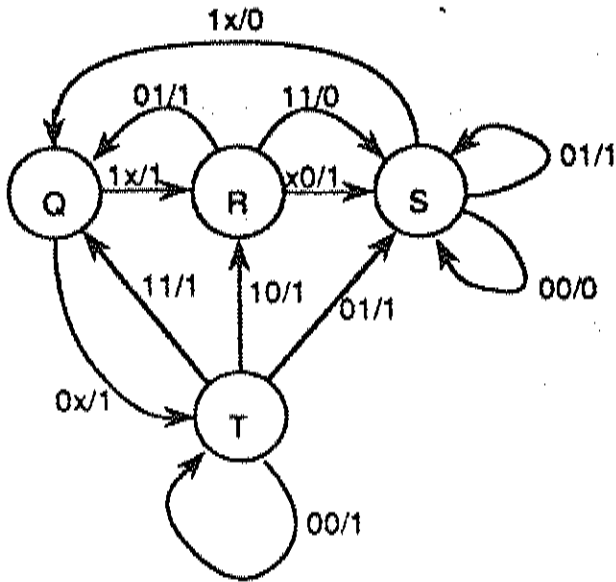
Go for it, Sidney! You've got it! You've got it!
Good hands! Don't choke!

MISTER GOFFO, by Joe Martin



2 Convert this MEALY diagram (same notation as in problem 1) into the simplest equivalent MOORE machine state diagram.

(10 min)



3

To make self-checking possible, fault-tolerant systems sometimes use "dual-rail" encoding in which every bit uses two lines and is represented with complimentary values on these lines. Thus each bit is encoded on the two lines as follows:

01 represents FALSE; 10 represents TRUE;

the remaining two codes (00 and 11) are illegal and indicate some error.

Assume that such an 8-bit dual-rail word is stored in a 16-bit register. Your task is to construct a fast dual-rail code checker for 8-bit parallel data, using only 2-input NAND gates. The output of this code checker should be one of the legal outputs (01 or 10) if all input bits are legal complementary values. If there is a single error, the output should be one of the illegal codes. Ignore multiple errors.

Use good hierarchical design techniques and construct:

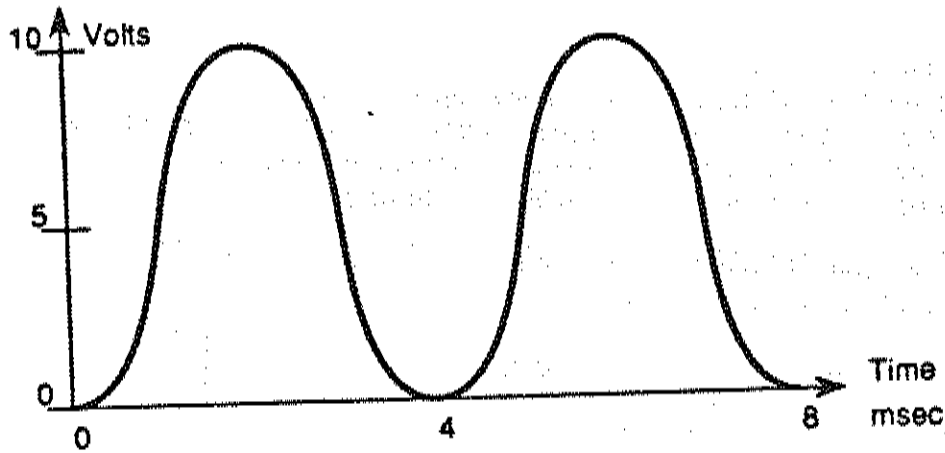
first a fast code checker macro block for two dual-rail bits having not more than

3 gate delays;

then use this macro block to build the 8-bit checker.

(20 min)

4



What is the dissipated power if the above (sine)-waveform is applied continuously to a microchip with an internal resistance of 20 Ohms ?

(10min)

What is the temperature of the microchip if it is encapsulated in a plastic package with thermal resistance of 100 degrees Celsius per Watt, if we assume that the surrounding room temperature is 20 degrees Celsius ? (You probably have never heard of "thermal resistance" ... But don't panic ! Use some common sense: how does the heat generated by the circuit get dissipated ? Then look at the dimension of the thermal resistance and do logical simple calculation.)

(4min)

5 Dr. X owns two Macintosh II computers that each need 4 memory boards in order to function. Both computers individually seem to develop a hardware problem that indicates that one of the boards in each machine is bad. Dr. X decides to combine boards from both computers to obtain at least one operational computer. He rips out all 8 boards and puts them onto a big unordered pile.

a) Dr. X picks four boards from this pile, inserts them into one of the two computers and boots the machine.

Assuming that there are exactly two bad boards in this pile, what are Dr. X's chances of hitting a working combination on his first try ?

(10min)

b) Could Dr. X have done better if he had not scrambled the boards coming from the two computers ? What strategy leads to the best odds of getting a working computer on the first try, if Dr. X turns on only one computer ?

What are those chances ?

(10min)

c) Since it takes a significant amount of time for the Macintosh to complete the boot process, is there an even better strategy that works in parallel with both computers ?

What are now the chances that either computer works on the first try ?
After how many "boot delays" is Dr. X guaranteed to have one working computer ?

(10min)

6

Consider 4 ways to provide a subroutine call capability within an instruction set:
 #1: The PC can be loaded from or copied into any general-purpose register. There is no other way to access the PC except through sequential execution or conventional JUMP instructions.

#2: A hardware stack of fixed length is provided in which the current PC is logically the top of stack. A new PC value can be set by a PUSH while a previously stacked value can be restored by a POP. Except for the top of stack, no other elements can be accessed. A "stack-full" signal is available.

#3: As in #2, except the bottom of the stack is accessible through a register named Q. Values in Q can be copied to / from memory using conventional register load/store instructions.

#4: The PC is saved in memory location 4 on every CALL, and restored from memory location 4 during RETURN.

For each technique answer the following questions:

- Describe all actions necessary to manipulate PC values to effect subroutine calls and returns. Consider nonreentrant, reentrant, and recursive routines. If the implementation requires software support, mention in your description the activities relegated to the software.
- Are any restrictions imposed on potential subroutine structures by the option being discussed? What are they?
- Discuss one advantage and one disadvantage of the option.

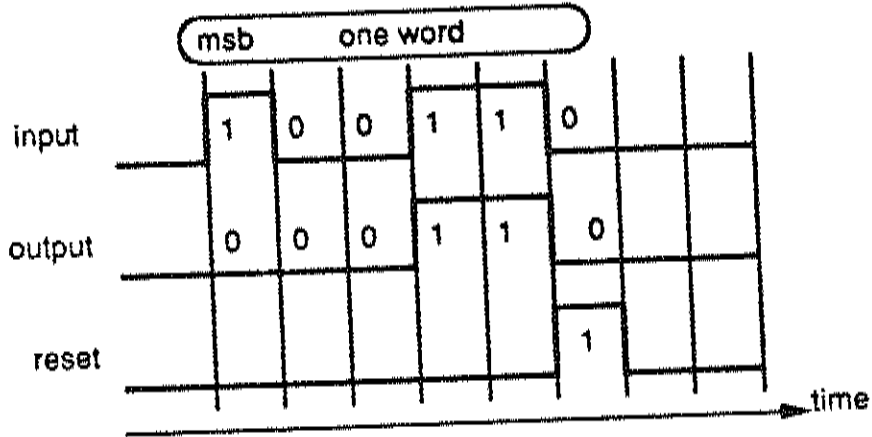
(24 min)

Hardware Prelim Spring 1989

6cont

- 7 Design a synchronous MEALY machine that takes a serial bit-stream (msb first) of a number of undetermined length and serially outputs a binary number representing the input divided by 5 using the normal "long division", i.e. the output is the number seen so far div 5. (HINT: use remainder as a state ...)
 The process starts from a reset state. When the reset input signal becomes TRUE, the output bit stream terminates and the machine returns to the reset state.

EXAMPLE:



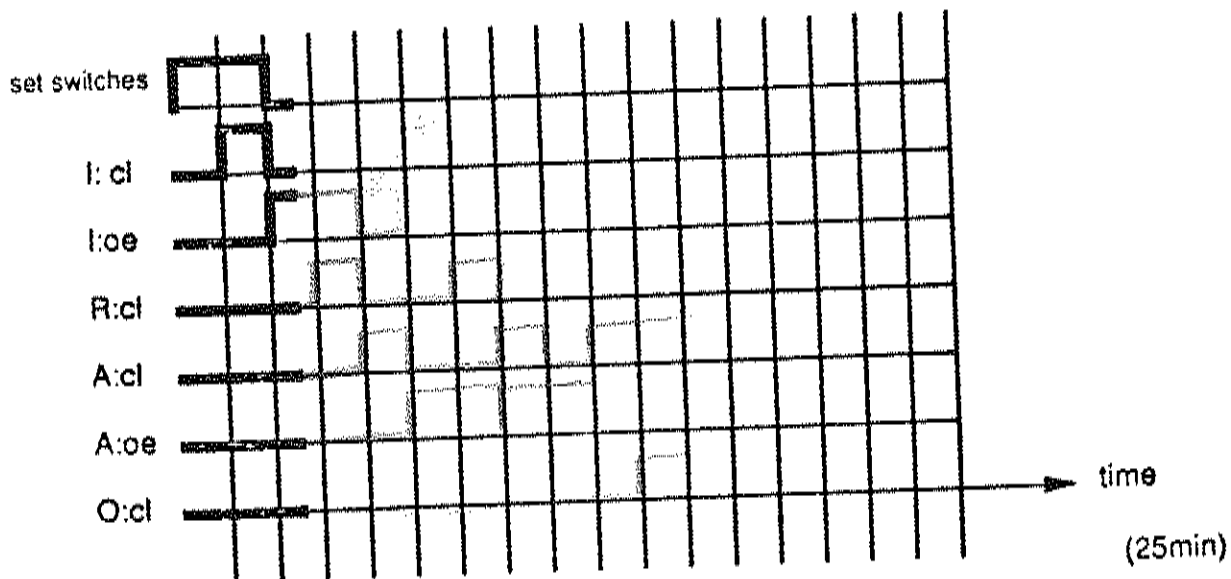
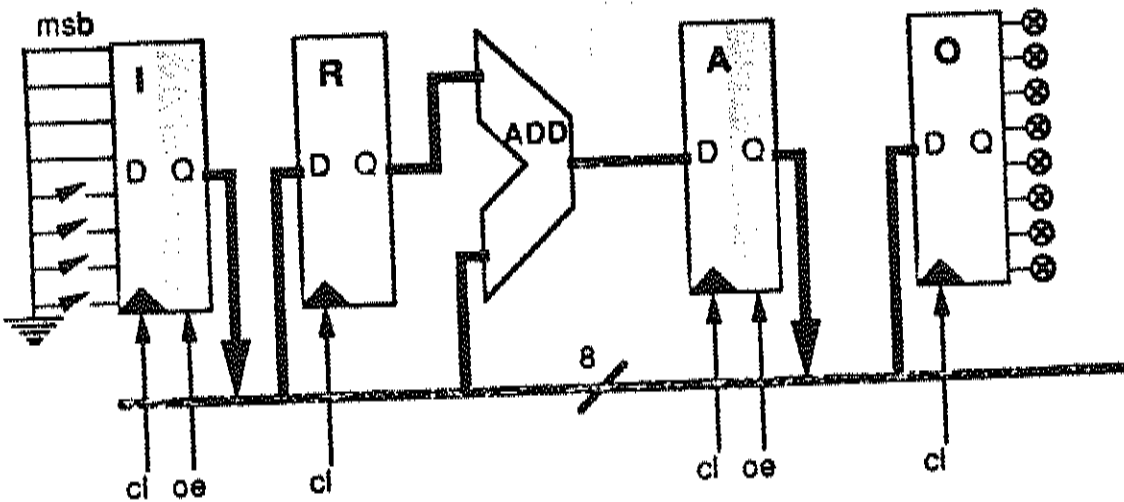
Draw a minimum state diagram, using the same notation as in problem 1.

(15 min)

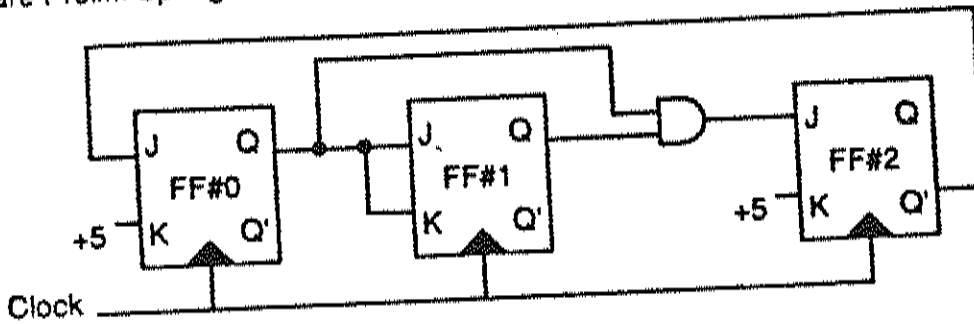
8 Assume that you have a bus-connected assembly of an adder and four registers as shown below. All registers are positive edge triggered, and registers I and A have tri-state output. All busses are 8 bits wide.

You want to multiply by 6 a number i that you set with the toggle switches attached to the input register I, and then display the result on the LEDs attached to the output register O.

Specify a minimal sequence of appropriate operations for the control signals cl (=clock input) and oe (=output enable) by completing the timing diagram below. Assume that the input switches have already been set for the input number (i).



9



Timing Analysis:

- a) All flipflops have a maximum delay of 60ns, a minimum delay of 30ns, a setup time of 10ns, and a hold time of 10ns. The AND gate has a maximum delay of 10ns and a minimum delay of 5ns. What is the maximum clock frequency at which the circuit can be operated properly? Give a simple timing diagram to justify your answer.

(7min)

- b) Assume the same specs as in (a) above. If FF#0 and FF#1 are clocked simultaneously, what is the maximum clock skew (before (-) and after (+)) for FF#2 for correct operation at 10 MHz? Again show a simple timing diagram.

(10min)

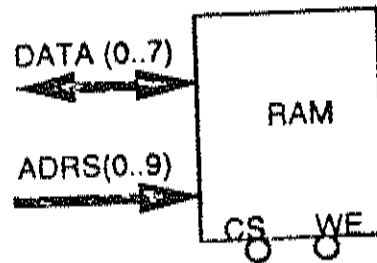
10

Design a synchronous First-In-First-Out Buffer (FIFO), with 1k characters of 8 bits each. When the FIFO is not empty, data is dumped at the rate of 10 characters/sec with no hand-shaking. The FIFO buffer should accept input until the internal buffer is full, then assert the NOT_READY output back to the computer. Handshaking on input consists of NEW_DATA_READY to FIFO, and ACK back to the source:



a) Give a block diagram implementation of such a device using MSI components: Registers, Counters, FlipFlops, Multiplexors, Adders, Comparators, etc, and a 1k RAM. Choose a straight-forward approach; no need to be fancy or to go into any details, e.g., you may just show one block for a "control FSM".

(18 min)



10cont

b) Show a timing diagram for the external FIFO signals for normal operation and when the FIFO is just getting full.

(10 min)

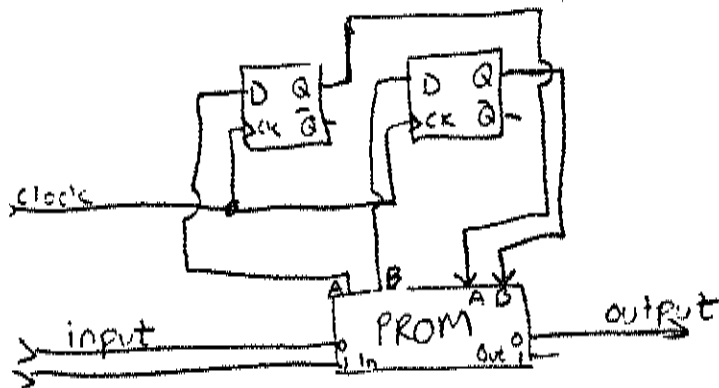
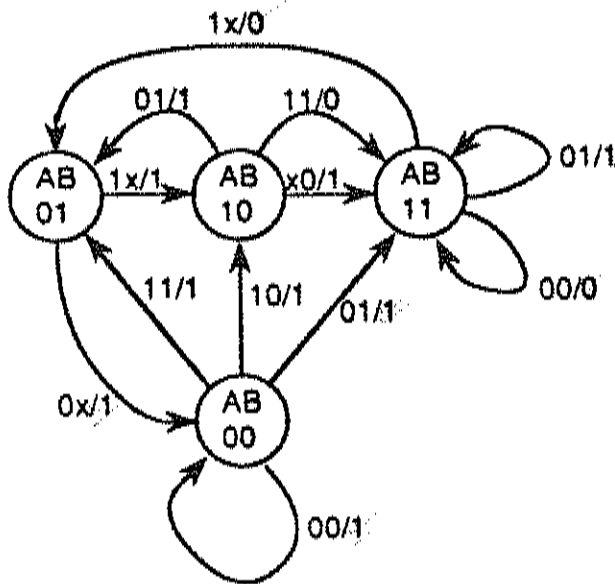
c) Describe how you detect overflow and underflow.

(2 min)

1

Below is the state diagram of a clocked MEALY machine. The notation {In}/{Out} on the transition arrows indicates how the machine's outputs {Out} react to a given set of inputs {In} while the machine is in the state from which the arrow emerges. The arrow itself indicates the transition taken by the machine if the clock occurs while the set of inputs is {In}.
 Using the given state assignment and the given PROM structure, design a MEALY-type controller with D-type registers of suitable size. Show a block diagram of your controller. List contents of the PROM as far as it is used.

(15 min)



We don't need O_1 .

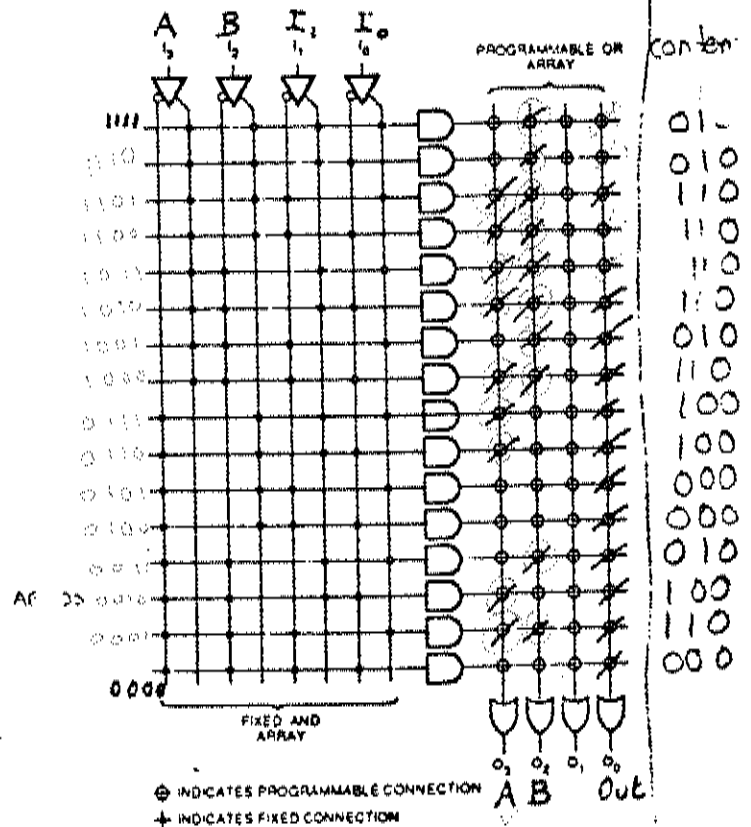
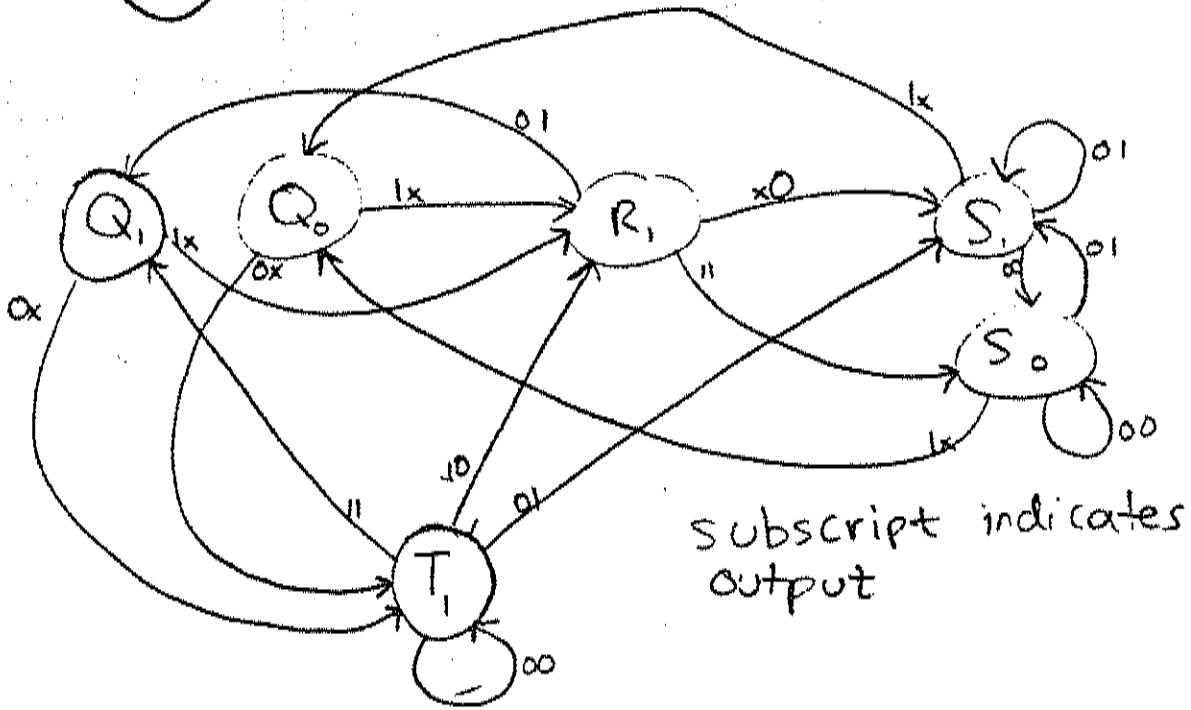
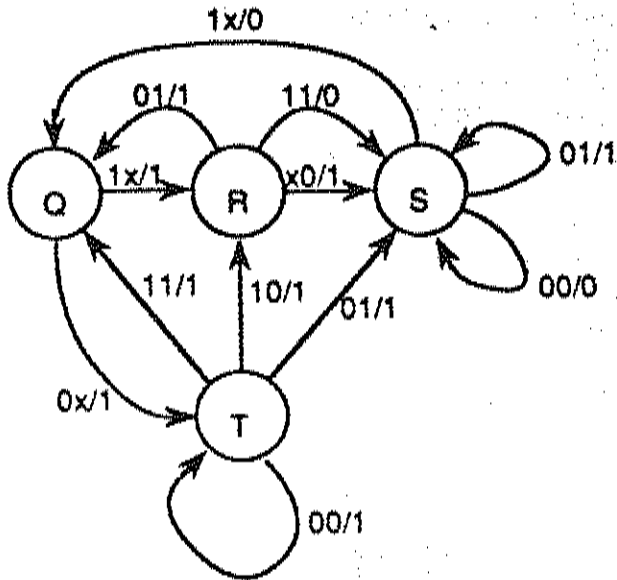


Figure 1. PROM Array Structure

2

Convert this MEALY diagram (same notation as in problem 1) into the simplest equivalent MOORE machine state diagram.

(10 min)



3 To make self-checking possible, fault-tolerant systems sometimes use "dual-rail" encoding in which every bit uses two lines and is represented with complimentary values on these lines. Thus each bit is encoded on the two lines as follows:

01 represents FALSE; 10 represents TRUE;

the remaining two codes (00 and 11) are illegal and indicate some error.

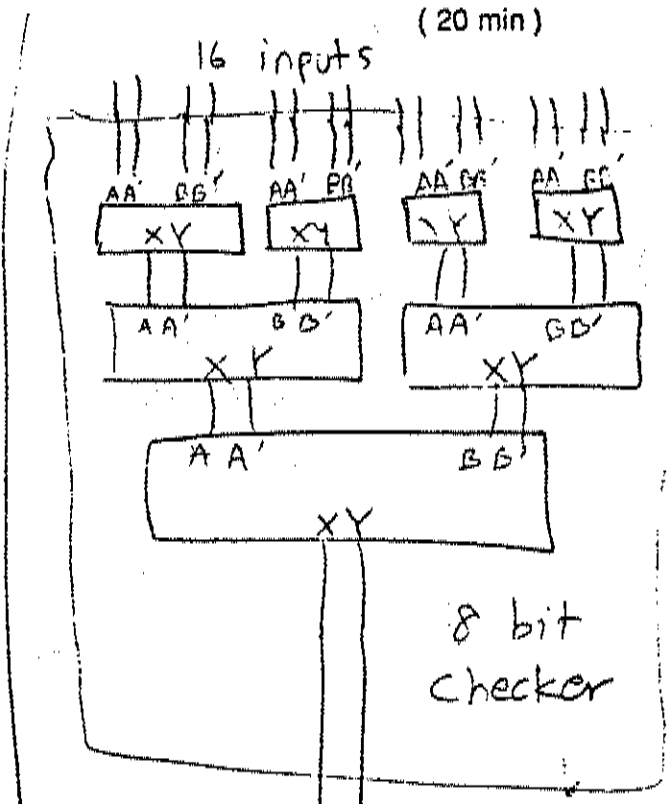
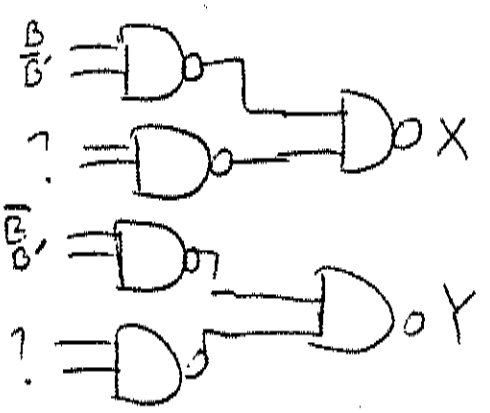
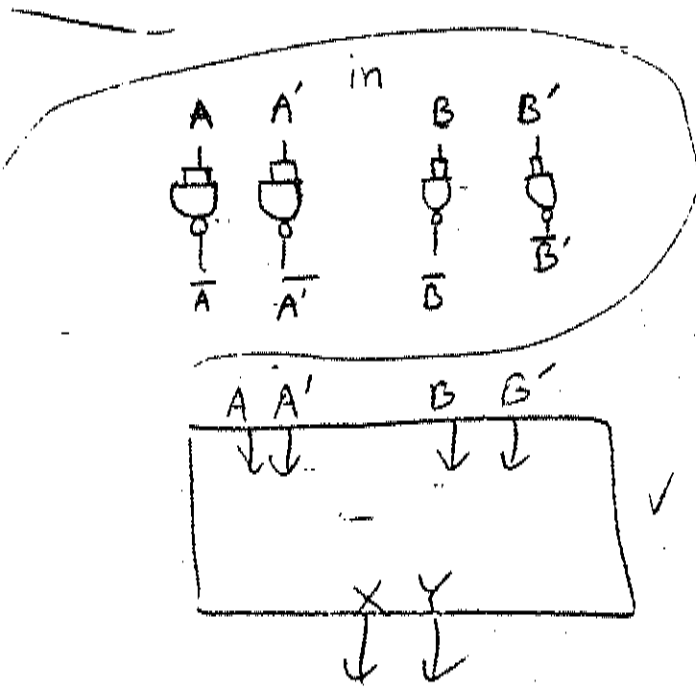
Assume that such an 8-bit dual-rail word is stored in a 16-bit register. Your task is to construct a fast dual-rail code checker for 8-bit parallel data, using only 2-input NAND gates. The output of this code checker should be one of the legal outputs (01 or 10) if all input bits are legal complementary values. If there is a single error, the output should be one of the illegal codes. Ignore multiple errors.

Use good hierarchical design techniques and construct:

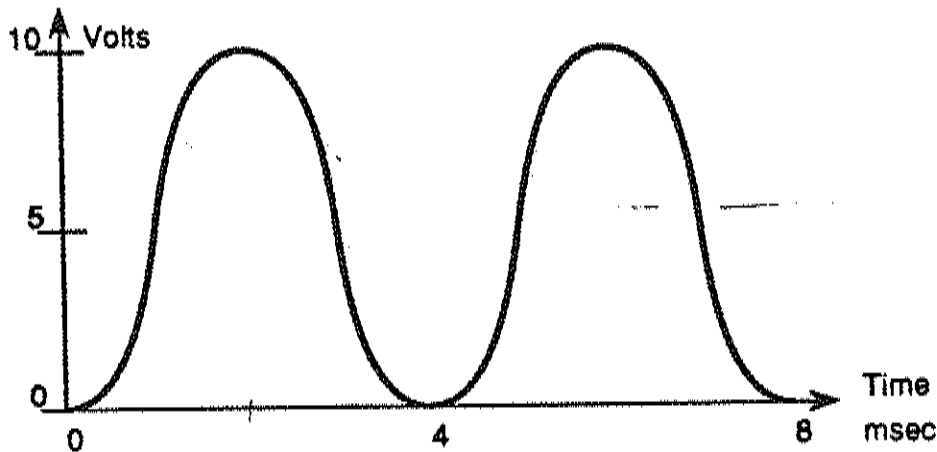
first a fast code checker macro block for two dual-rail bits having not more than

3 gate delays;

then use this macro block to build the 8-bit checker.



4



What is the dissipated power if the above (sine)-waveform is applied continuously to a microchip with an internal resistance of 20 Ohms?

(10min)

$$P = IV = V^2/R$$

$$\frac{1}{4R} \int_0^4 \left(5 \sin \frac{\pi x}{4} + 5 \right)^2 dx$$

$$P = \frac{1}{4R} \left[\int_0^4 25 \sin^2 \frac{\pi x}{4} dx + \int_0^4 50 \sin \frac{\pi x}{4} dx + 100 \right] \quad \text{where } R=20$$

What is the temperature of the microchip if it is encapsulated in a plastic package with thermal resistance of 100 degrees Celsius per Watt, if we assume that the surrounding room temperature is 20 degrees Celsius? (You probably have never heard of "thermal resistance" ... But don't panic! Use some common sense: how does the heat generated by the circuit get dissipated? Then look at the dimension of the thermal resistance and do logical simple calculation.)

(4min)

$$\Delta T = RP$$

$$T = 20 + 100P \quad \text{where } P \text{ is from above}$$

5 Dr. X owns two Macintosh II computers that each need 4 memory boards in order to function. Both computers individually seem to develop a hardware problem that indicates that one of the boards in each machine is bad. Dr. X decides to combine boards from both computers to obtain at least one operational computer. He rips out all 8 boards and puts them onto a big unordered pile.

a) Dr. X picks four boards from this pile, inserts them into one of the two computers and boots the machine.
 Assuming that there are exactly two bad boards in this pile, what are Dr. X's chances of hitting a working combination on his first try?

$\frac{3}{14}$

(10min)

$$P(\text{all good boards}) = \frac{6}{8} \cdot \frac{5}{7} \cdot \frac{4}{6} \cdot \frac{3}{5} = \frac{3}{14}$$

✓

b) Could Dr. X have done better if he had not scrambled the boards coming from the two computers? What strategy leads to the best odds of getting a working computer on the first try, if Dr. X turns on only one computer?

Choose 2 boards from each computer
 $P(\text{works}) = \left(\frac{3}{4} \cdot \frac{2}{3}\right) \cdot \left(\frac{3}{4} \cdot \frac{2}{3}\right) = 1/4$

$\frac{1}{4}$

(10min)

c) Since it takes a significant amount of time for the Macintosh to complete the boot process, is there an even better strategy that works in parallel with both computers?

① Split Two from each computer
 ② If good, put sets x & z together and sets y & w together
 Please see back of previous page
 What are now the chances that either computer works on the first try?
 After how many "boot delays" is Dr. X guaranteed to have one working computer?

$\frac{1}{2}$

2

(10min)

5

Dr. X owns two Macintosh II computers that each need 4 memory boards in order to function. Both computers individually seem to develop a hardware problem that indicates that one of the boards in each machine is bad. Dr. X decides to combine boards from both computers to obtain at least one operational computer. He rips out all 8 boards and puts them onto a big (unordered pile).

a) Dr. X picks four boards from this pile, inserts them into one of the two computers and boots the machine.

Assuming that there are exactly two bad boards in (this pile), what are Dr. X's chances of hitting a working combination on his first try?

$\frac{3}{14}$

(10min)

$$\frac{6}{8} \cdot \frac{5}{7} \cdot \frac{4}{6} \cdot \frac{3}{5} = \frac{3}{14}$$

(Still assuming just 2 out of 8 are bad)

b) Could Dr. X have done better if he had not scrambled the boards coming from the two computers? What strategy leads to the best odds of getting a working computer on the first try, if Dr. X turns on only one computer?

pull two out of one computer + replace w/ two from the other.

$$\frac{3}{4} \cdot \frac{2}{3} \cdot \frac{3}{4} \cdot \frac{2}{3} = \frac{4}{16} = \frac{1}{4}$$

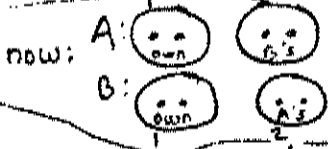
$\frac{1}{4}$

(10min)

What are those chances?

c) Since it takes a significant amount of time for the Macintosh to complete the boot process, is there an even better strategy that works in parallel with both computers?

Computers A & B, swap 2 board in A & B (now has 2 of the others)
 • boot both
 • if failed try set A1 & B1 in A
 + A2 & B2 in B
 • boot both
 • if failed place A1 & B2 in A
 A2 & B1 in B — must work



8 choose 4

2 bad out of

$$\frac{6}{12} \cdot \frac{6}{12} + \frac{6}{12} \cdot \frac{6}{12} = \frac{1}{2}$$

What are now the chances that either computer works on the first try?

$\frac{1}{2}$

After how many "boot delays" is Dr. X guaranteed to have one working computer?

3

(10min)

2E

21

6

- Consider 4 ways to provide a subroutine call capability within an instruction set:
- #1: The PC can be loaded from or copied into any general-purpose register. There is no other way to access the PC except through sequential execution or conventional JUMP instructions.
 - #2: A hardware stack of fixed length is provided in which the current PC is logically the top of stack. A new PC value can be set by a PUSH while a previously stacked value can be restored by a POP. Except for the top of stack, no other elements can be accessed. A "stack-full" signal is available.
 - #3: As in #2, except the bottom of the stack is accessible through a register named Q. Values in Q can be copied to / from memory using conventional register load/store instructions.
 - #4: The PC is saved in memory location 4 on every CALL, and restored from memory location 4 during RETURN.

For each technique answer the following questions:

- (a) Describe all actions necessary to manipulate PC values to effect subroutine calls and returns. Consider nonreentrant, reentrant, and recursive routines. If the implementation requires software support, mention in your description the activities relegated to the software.
- (b) Are any restrictions imposed on potential subroutine structures by the option being discussed? What are they?
- (c) Discuss one advantage and one disadvantage of the option.

(24 min)

#1 (a) nonreentrant: 1 return storage used/routine
 call: store pc there
 jump
 ret: pc ← word
 reentrant/recursive: push pc on stack
 call: reg ← pc
 add reg, 2
 push reg
 ret: pop into reg
 pc ← reg
 reg ← pc
 add reg, 2
 word ← reg
 pc ← sub addr
 OK

(b) if we can't have a stack we can't have reentrant/recursive code. but this can be built by software

(c) advantage: simple hardware for jsr, ret
 dis: requires explicit stack maintenance for recursion/reentrancy

#2 (a) use stack
 call: push pc (jsr)
 ret: pop pc (ret)
 (full stack causes trap)

6cont

(b) - the subroutine must leave the stack in the same state on exit.
(not a big deal).

(c) +: reentrancy / recursion is easy
-: stack is in hardware w/ limited size
(limits depth of recursion)

#3 same comments as #2

I don't see why the bottom is useful; except that we can store there the addr of an error routine to trap:

the case of a pop (return) when the stack is empty

allows you to extend fixed size stack to an in-memory stack managed by SW.

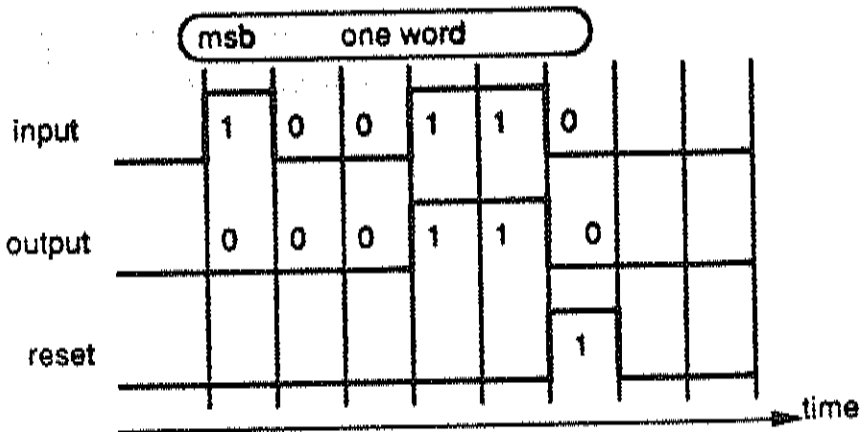
#4 (a) general case for recursion.
call: push [4] on stack
call subr
pop [4]
ret: ret

(b) subroutine must leave stack in same state on exit.

(c) +: simple hardware
-: must manually maintain stack

- 7 Design a synchronous MEALY machine that takes a serial bit-stream (msb first) of a number of undetermined length and serially outputs a binary number representing the input divided by 5 using the normal "long division", i.e. the output is the number seen so far div 5. (HINT: use remainder as a state ...)
 The process starts from a reset state. When the reset input signal becomes TRUE, the output bit stream terminates and the machine returns to the reset state.

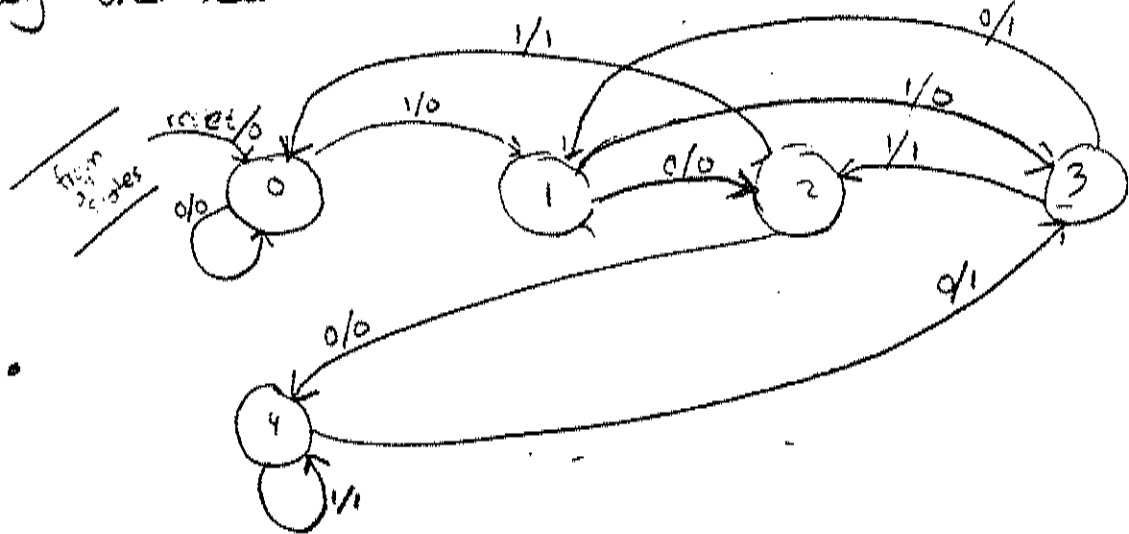
EXAMPLE:



Draw a minimum state diagram, using the same notation as in problem 1.

only the last 4 bits concern us

(15 min) / 5

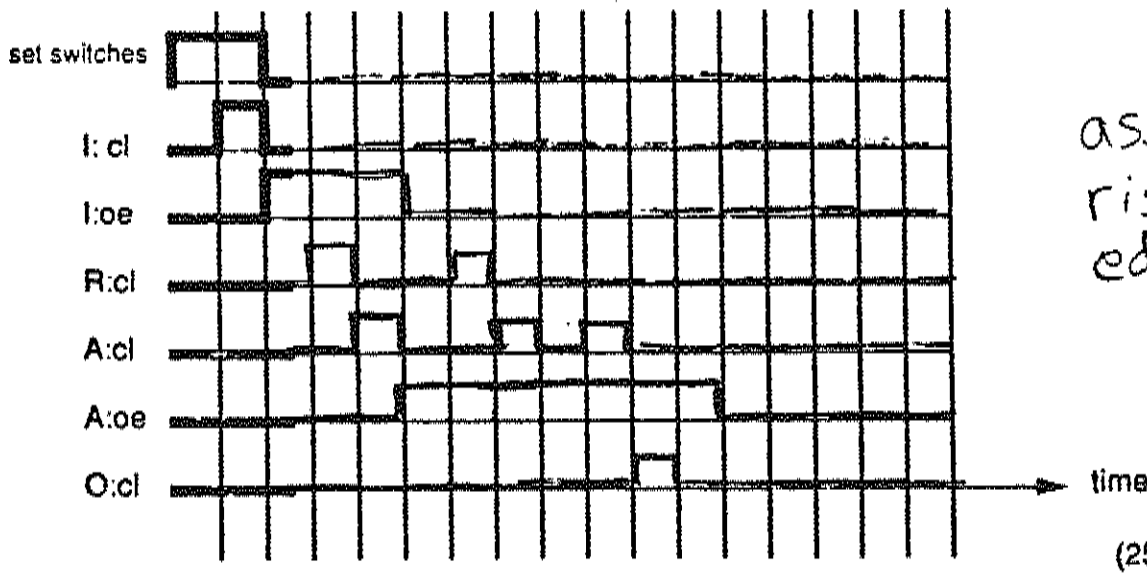
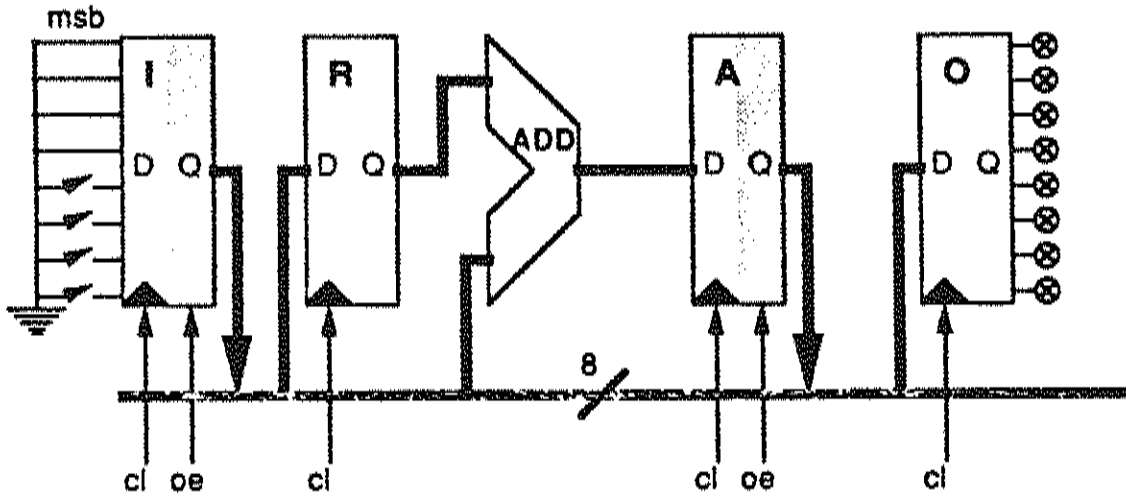


*w/ each new bit,
 0: remainder doubles
 1: remainder doubles & increments
 when $rem \geq 5$, send 1, else 0*

8 Assume that you have a bus-connected assembly of an adder and four registers as shown below. All registers are positive edge triggered, and registers I and A have tri-state output. All busses are 8 bits wide.

You want to multiply by 6 a number i that you set with the toggle switches attached to the input register I, and then display the result on the LEDs attached to the output register O.

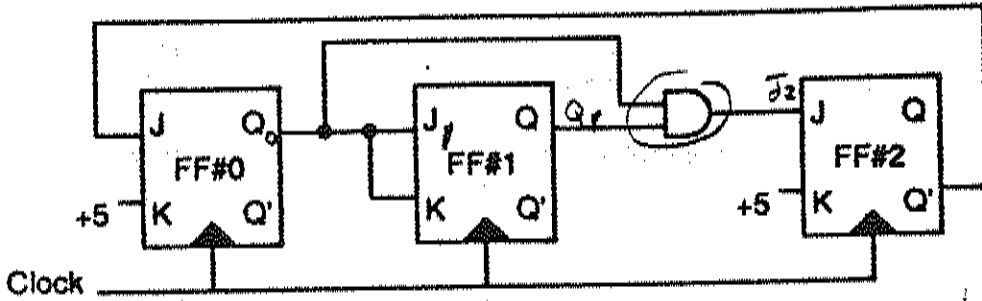
Specify a minimal sequence of appropriate operations for the control signals cl (=clock input) and oe (=output enable) by completing the timing diagram below. Assume that the input switches have already been set for the input number (i).



$A = 2x$
 $A = 4x$
 $A = 6x$

Adnan

9

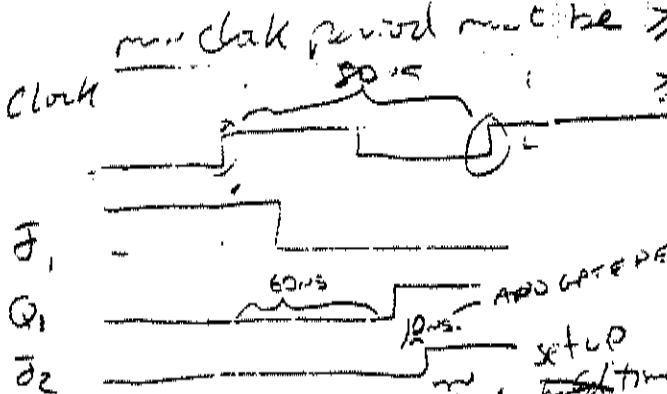


P 10/7

Timing Analysis:

- a) All flip-flops have a maximum delay of 60ns, a minimum delay of 30ns, a setup time of 10ns, and a hold time of 10ns. The AND gate has a maximum delay of 10ns and a minimum delay of 5ns. What is the maximum clock frequency at which the circuit can be operated properly? Give a simple timing diagram to justify your answer.

Assume + ideg trigger ed FF's ✓



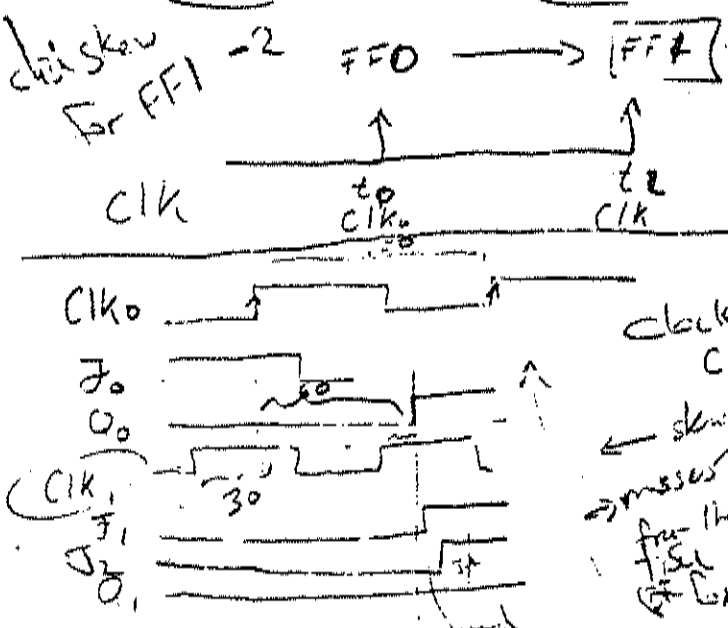
max clock period must be \geq max FF delay + max gate delay + 10 (7min)

$80 \text{ ns} \geq 60 \text{ ns} + 10 \text{ ns} + 10 \text{ ns}$

\Rightarrow max frequency is $\frac{1}{80 \text{ ns}}$

$\frac{1}{80 \times 10^{-9}} = 12.5 \text{ MHz}$

- b) Assume the same specs as in (a) above. If FF#0 and FF#1 are clocked simultaneously, what is the maximum clock skew (before (-) and after (+)) for FF#2 for correct operation at 10 MHz? Again show a simple timing diagram.



clock skew = 100 ns

late max $t_2 - t_0 = 4.0 \text{ ns}$

early max $t_0 - t_2 = 100 - 60 \text{ ns} = 40 \text{ ns}$

30 ns delay for FF2

10 ns setup for FF2

10 ns hold for FF1

clock 2? C1 too fast early misses the input for the data 30 ns for FF1?

hold time? -3

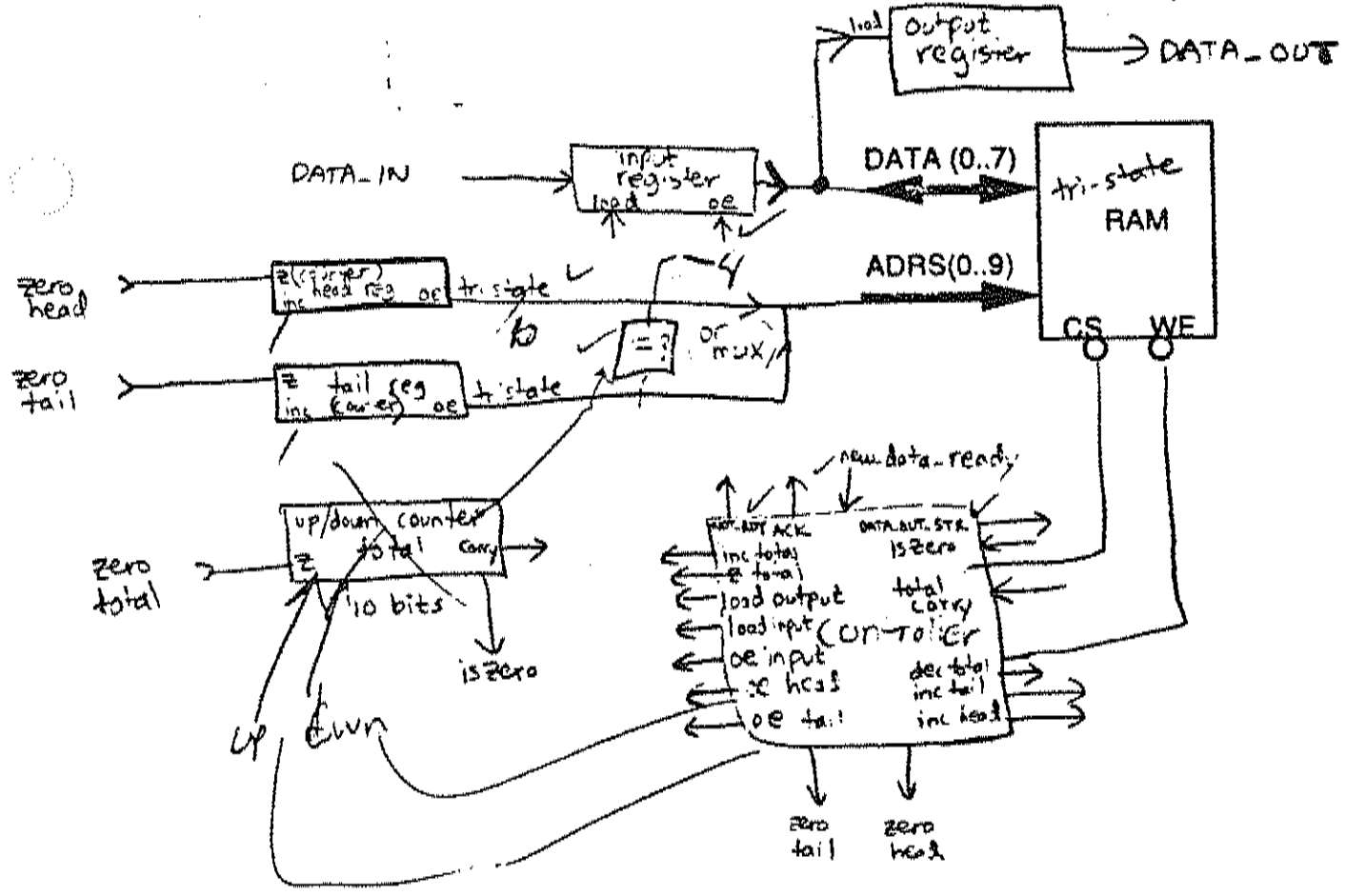
rise condition for CLK2 is too slow

10 Design a synchronous First-In-First-Out Buffer (FIFO), with 1k characters of 8 bits each. When the FIFO is not empty, data is dumped at the rate of 10 characters/sec with no hand-shaking. The FIFO buffer should accept input until the internal buffer is full, then assert the NOT_READY output back to the computer. Handshaking on input consists of NEW_DATA_READY to FIFO, and ACK back to the source:



a) Give a block diagram implementation of such a device using MSI components: Registers, Counters, FlipFlops, Multiplexors, Adders, Comparators, etc, and a 1k RAM. Choose a straight-forward approach; no need to be fancy or to go into any details, e.g., you may just show one block for a "control FSM".

(18 min)



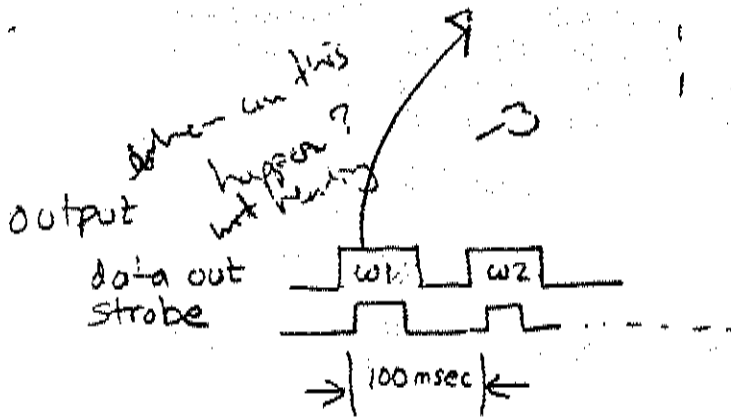
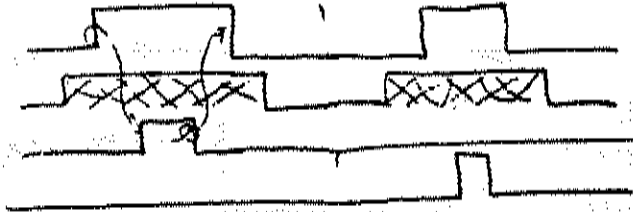
10cont

b) Show a timing diagram for the external FIFO signals for normal operation and when the FIFO is just getting full.

(10 min)

loading a number

new data ready
Data in ack
not-ready



1024th word

c) Describe how you detect overflow and underflow.

(2 min)

the up/down counter total records how many entries we have — if its zero the FIFO is empty — if its 1024, the FIFO is full & the carry out signal tells us. it is adjusted w/ each change.

*up on write
down on read?*