Department of Electrical and Computer Engineering
The University of Texas at Austin

EE 460N Spring 2017
Y. N. Patt, Instructor
Chirag Sakhuja, Sarbartha Banerjee, Jonathan Dahm, Arjun Teh, TAs
Exam 1
March 1, 2017

Name: Jon Dahm

Problem 1 (25 points):_____

Problem 2 (10 points):_____

Problem 3 (15 points):_____

Problem 4 (25 points):_____

Problem 5 (25 points):_____

Total (100 points):_____

Note: Please be sure that your answers to all questions (and all supporting work that is required) are contained in the space provided.

Note: Please be sure your name is recorded on each sheet of the exam.

Please sign the following. I have not given nor received any unauthorized help on this exam.

Signature: lol nope

GOOD LUCK!

Name: Jon Dahm

Problem 1 (25 points): Answer any five of the six. Draw a line through the one you do not want graded.

Part a (5 points): A load-store ISA does not allow what?

Operations on values in memory

Part b (5 points): Unaligned accesses. Part of the ISA or part of the microarchitecture? Explain.

ISA

Part c (5 points): Interleaving. Part of the ISA or part of the microarchitecture? Explain.

μArch

Part d (5 points): J.E.Smith's branch predictor introduced the use of saturating 2-bit counters. What does the word "saturating" mean in this context, and why is it necessary?

11 + 1 = 11
00 − 1 = 00

Prevent overflows, which would lead to mispredictions

Part e (5 points): McFarling modified my GAs predictor, creating g-share. What was his purpose for doing so?

To try to reduce interference between branches.
(XORs branch address w/ BHT)

Part f (5 points): Do processes having higher priority get increased privilege? If yes, explain why. If no, explain why not.

No.

Name: Jon Dyhm

Problem 2 (10 points): An Aggie hates the LC-3b, so he cuts the 16 wires labeled A and B on the data path, and grounds the wire labeled C in the microsequencer so the LC-3b can not function properly. (The data path, showing wires A and B is shown on the next page.)

Part a (2 points): If the ~~16~~ 8 wires A are cut, which instruction(s) are impossible to function? Explain.

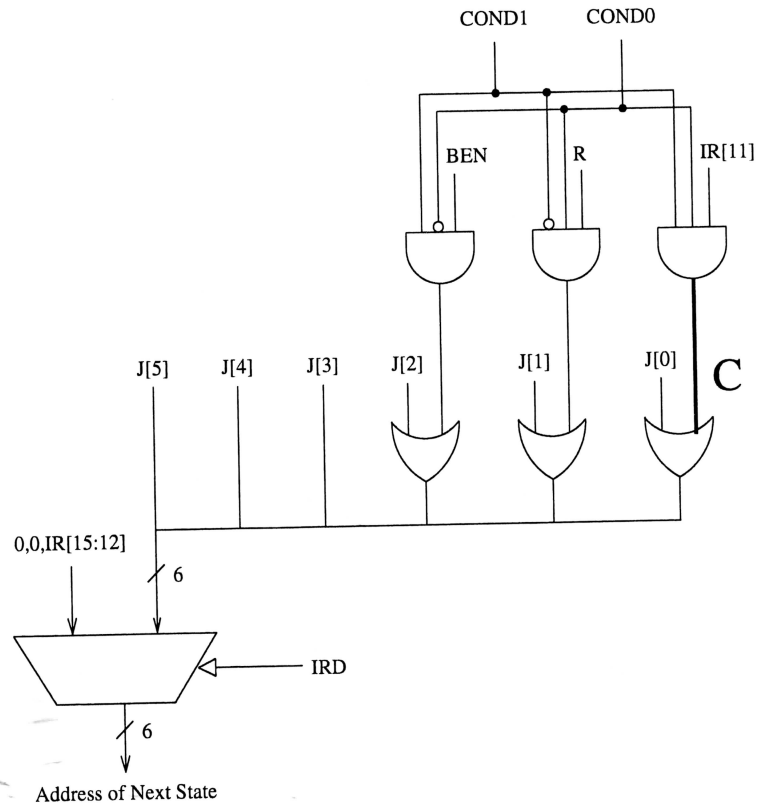
Instruction(s)	Explanation
TRAP	No way to get trap vector onto the bus.

Part b (4 points): If the 16 wires B are cut, which instruction(s) are impossible to function? Explain.

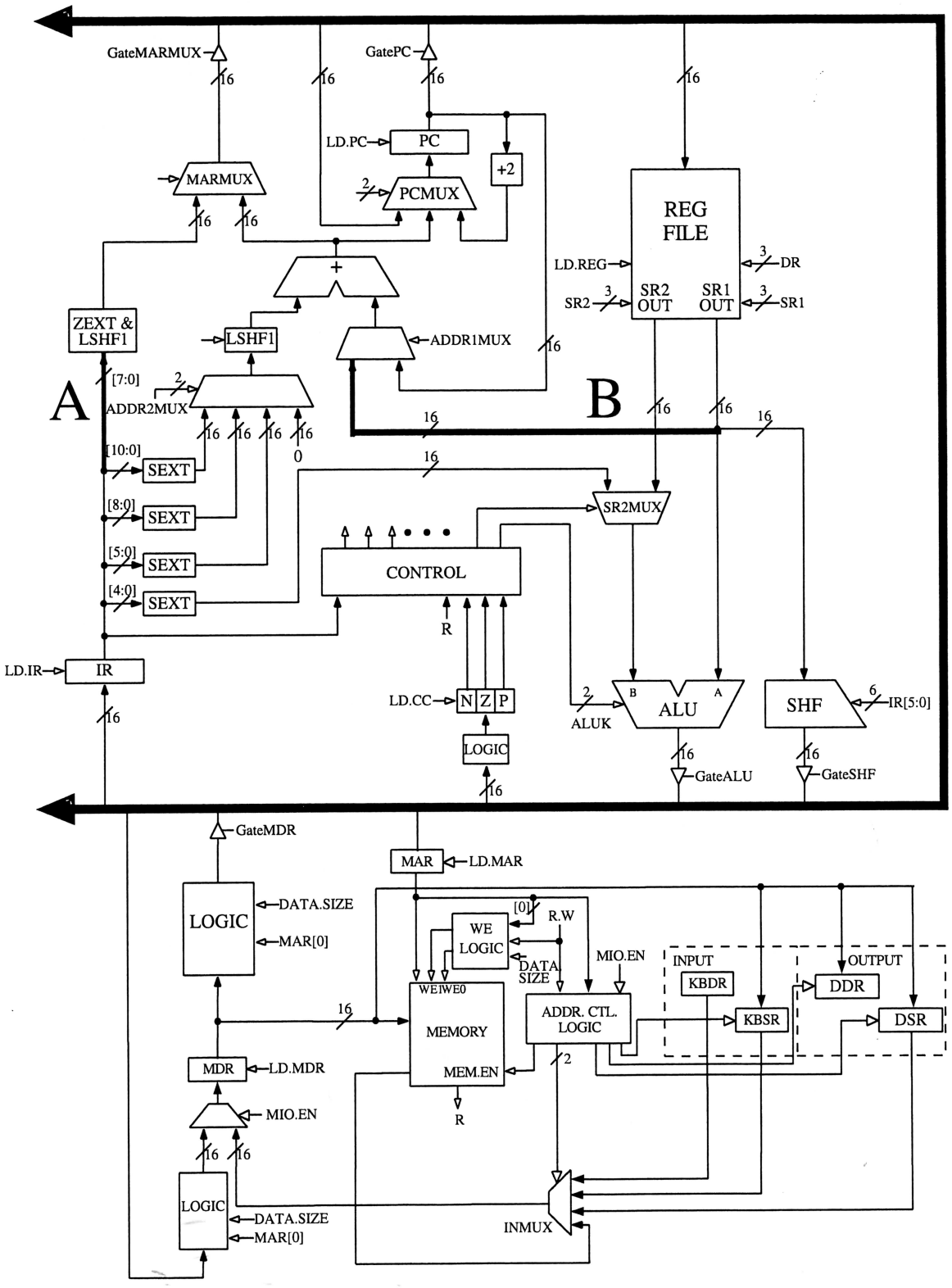
Instruction(s)	Explanation
LDB LDW STB STW	These instructions require a base register plus an immediate offset, which requires the base register value to pass through B.

Part c (4 points): If wire C is grounded, which instruction(s) are impossible to function? Explain.

Instruction(s)	Explanation
JSR(R)	With C grounded, there's no longer any way for the machine to distinguish between JSR and JSRR, so it will default to JSRR.



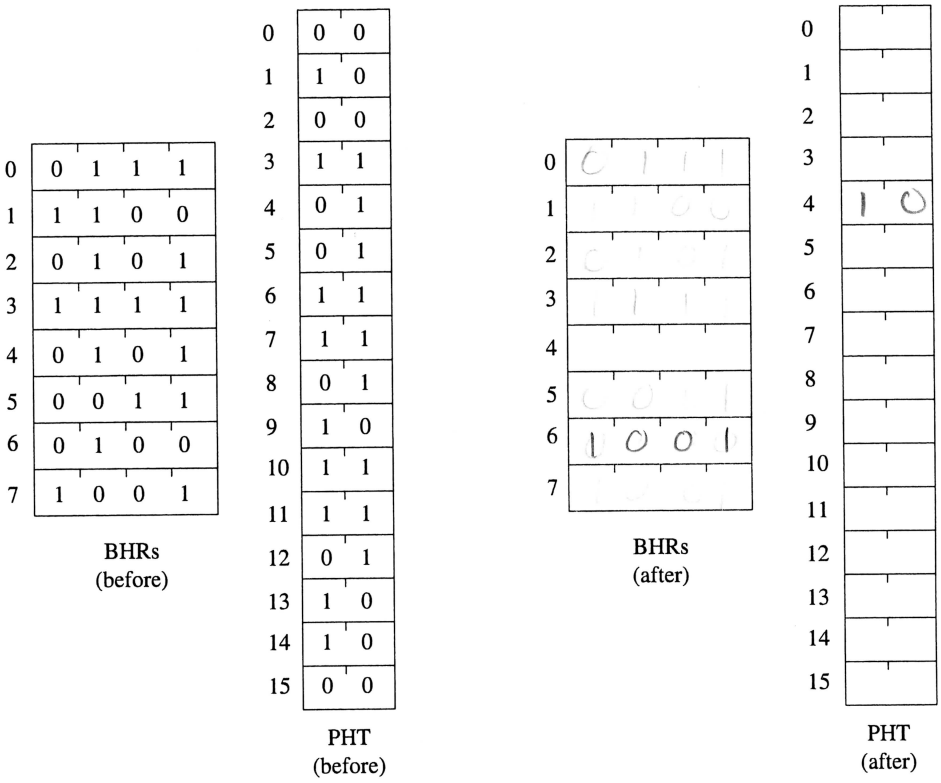
Name: Jon Dahm



Name: Jon Dahm

Problem 3 (15 points): In this problem we add an SAg 2-level branch predictor to the LC-3b. Recall that the S means the branches are partitioned into sets, where all branches in a set share the same BHR. In our case, we have 8 sets, and therefore 8 BHRs. Bits 13, 10, and 7 of a branch's address determine which set a branch belongs to. For example, a branch at address x9E84 uses BHR 3 because bit 13 is 0, and bits 10 and 7 are 1.

The current state of the BHRs are shown below. The direction (taken = 1, not taken = 0) of the most recent branch is the right-most bit of its respective BHR.



x360E
= 0011 0110 0000 1110
↑ ↑ ↑

Part a (5 points): The PC contains x360E, and the instruction fetched is a branch. Does the branch predictor predict taken or not taken? On what information is your answer based? Please be specific.

x360E → BHR[6]
BHR[6] = 0100₂ = 4
PHT[4] = 01

Predict not taken

Part b (5 points): The branch at x360E is taken. Compute the new BHRs and PHT in the figure above after this branch completes execution. It is only necessary to show the "after" entries that have changed.

PROBLEM CONTINUTED ON NEXT PAGE

Name: Jon Dahm

Part c (5 points): Execution of the program results in four more branches (for a total of five) being executed. They are at locations xCC48, xD028, x4842, and x6974. Each of the five branches retires before the next branch is fetched. The table below shows the prediction and direction for each of these five branches. Your job: complete the table below. Do not make any changes to the figures on the previous page.

Branch at address	Prediction	Actual
x360E	(Answer in part a)	Taken
xCC48	Not Taken	Taken
xD028	Taken	Taken
x4842	Not Taken	Not Taken
x6974	Taken	Not Taken

xCC48 → BHR[2] = 0101 = 5 PHT[5] = 01

= 1100 1100 0100 1000

↑ ↑ ↑

xD028 → BHR[0] = 0111 = 7 PHT[7] = 11

= 1101 0000 0010 1000

↑ ↑ ↑

x4842 → BHR[0] = 1111 = 15 PHT[15] = 00

= 0100 1000 0100 0010

↑ ↑ ↑

x6974 → BHR[4] = 0101 = 5 PHT[5] = 10

= 0110 1001 0111 0100

↑ ↑ ↑

Name: Jon Dahm

Problem 4 (25 points): An out-of-order processor executes its instructions according to the Tomasulo algorithm. The ISA specifies 8 registers, R0 to R7. The microarchitecture contains one pipelined adder and one pipelined multiplier. Pipelining allows an add (or multiply) instruction to initiate execution each clock cycle.

- Fetch and Decode take one cycle each.
- ADD execution takes 3 cycles
- MUL execution takes 5 cycles.
- For ADD and MUL, if two instructions are ready to dispatch to the same functional unit in the same cycle, the older instruction is dispatched and the younger instruction waits.
- For ADD and MUL, one cycle is needed to write the result to a destination register. Only one result can be written in a single clock cycle. If two instructions want to write results in the same clock cycle, the older instruction writes, and the younger is stored in a buffer and is available for writing in the following cycle.
- Data forwarding is not implemented.

The adder and multiplier each have 3-entry reservation stations. Note: if an instruction is of the form ADD Rx, Ry, Rz or MUL Rx, Ry, Rz, and Ry contains valid data 74 and Rz contains valid data 27, the reservation station entry has the form:

V	TAG	VALUE	V	TAG	VALUE
1	—	74	1	—	27

The reservation stations are initially empty and are filled from top to bottom. Each instruction remains in the reservation station until the end of the cycle in which it writes its result to a register.

The table below contains a program of seven instructions that are executed.

I1	ADD	R3 (5)	R1	R2
I2	MUL	R2	R1	R4
I3	MUL	R5 (6)	R3 (5)	R0
I4	ADD	R7	R1	R3
I5	ADD	R1	R1	R3
I6	MUL	R6	R2	R7
I7	ADD	R5	R3	R0

(1) (from reservation station)
(3) (from reservation station)
(2) (from reservation station)
(4) (from reservation station)
(8)
(10)
(9)

PROBLEM CONTINUTED ON NEXT PAGE

Name: Jon Dahm

Three snapshots of the machine are shown: (a) before execution, (b) after clock cycle 5, and (c) after clock cycle 9. Note that some information in the program (shown on the previous page), in the register file, and in the reservation stations are missing.

	V	TAG	VALUE
R0	1	—	0
R1	1	—	1
R2	1	—	2
R3	1	—	3
R4	1	—	4
R5	1	—	5
R6	1	—	6
R7	1	—	7

(a) Beginning

	V	TAG	VALUE
R0	1	—	0
R1	1	—	1
R2	0	δ	—
R3	0	α	—
R4	1	—	4
R5	0	σ	—
R6	1	—	6
R7	0	β	—

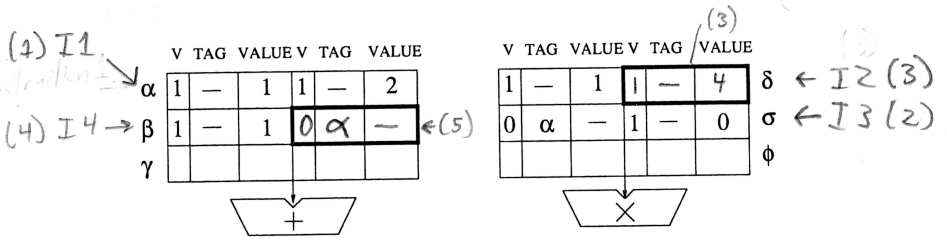
(3), (5), (6), (4)

(b) After cycle 5

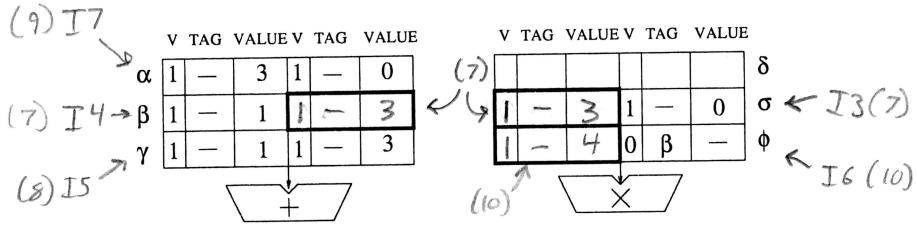
	V	TAG	VALUE
R0	1	—	0
R1	0	γ	—
R2	1	—	4
R3	1	—	3
R4	1	—	4
R5	0	α	—
R6	0	φ	—
R7	0	β	—

(8), (9), (10)

(c) After cycle 9



(b) After cycle 5



(c) After cycle 9

Part a (20 points): Your job: Fill in the missing information in the program (shown on the previous page), and in the bolded boxes in the register file and reservation stations at each of the snapshots.

Part b (5 points): The figure below shows, for each clock cycle, which phase of the instruction cycle each of the seven instructions are in. Complete the figure. We have provided 20 clock cycles. Only use what you need.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
I1	F	D	A	A	A	R3														
I2		F	D	M	M	M	M	M	R2											
I3			F	D	—	—	M	M	M	M	M	R5								
I4				F	D	—	A	A	A	R7										
I5					F	D	—	A	A	A	R1									
I6						F	D	—	—	—	M	M	M	M	M	R6				
I7							F	D	A	A	A	—	R5							

(1) The first instruction is ADD —, R1, R2, tag = α
- We know it's an add, so it must be the first thing in the adder's reservation station. The operands are values 1 and 2, which uniquely ID R1 + R2

(2) The third instruction is MUL —, α, R0, tag = σ
- The second entry of the multiplier's reservation station has a second operand of 0, which uniquely identifies R0. I2 and I4 both have specified second operands, and neither is R0, so I3 must be σ.

(3) The second instruction must be MUL R2, R1, R4, tag = δ
- If δ were I4, it would have to follow I3(σ) in the reservation station.
- The register file tells us that δ is written to R2.
- The register file tells us R4 hasn't been modified yet, so its value must be in reservation station.

(4) I4 must be ADD R7, R1, —, tag = β
- Process of elimination
- Reg file tells us β written to R7.

(5) I1(α) must write to R3. (∴ I3(σ) op1 = R3; I4(β) op2 is α)
- R3 is valid at the end of cycle 9; I3 could not have retired by then.

(6) I3(σ) must write to R5.
- Process of elimination.

(7) I3(σ) and I4(β) are still executing at the end of cycle 9.

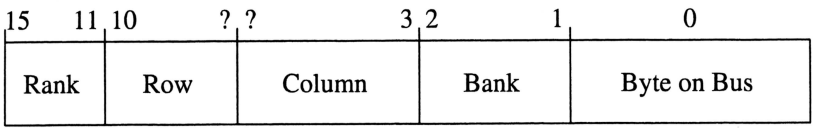
(8) I5 must be ADD R1, R1, R3, tag = γ
- γ is an adder tag. γ is written to R1.
- There are only two new ADD instructions processed, and one of them (I7) has a DR of R5, so it's not γ
- I6 has a first operand of R2, but γ doesn't reference R2.

(9) I7 must be ADD R5, R3, R0, tag = α

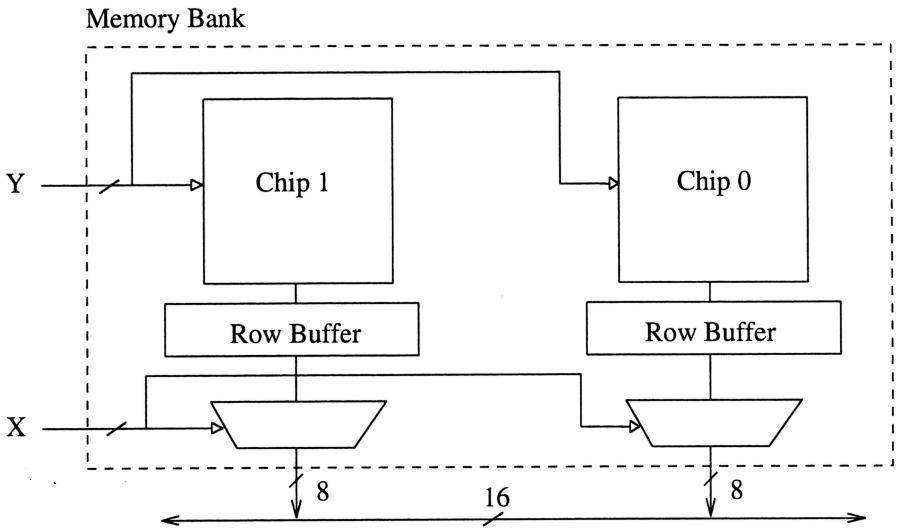
(10) I6 must be MUL R6, R2, R7, tag = φ
- Process of elimination
- β still references R7

Name: Jon Dahm

Problem 5 (25 points): Suppose we have a 4-way interleaved DRAM memory, with bits[2:1] designating the bank. All address bits are as shown. Note, the question marks below; in part b, you are going to have to determine how many row bits and how many column bits.



Recall that a single memory bank has the following form:



A row access takes 13 cycles, and a subsequent column access takes 3 cycles.

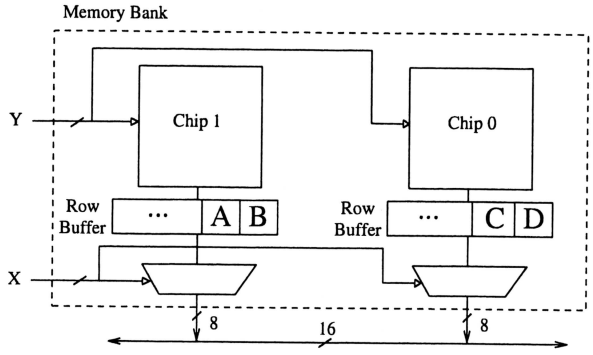
We store sequentially in this memory, starting at location x8000, an array of 64 English words, each containing 7 letters. Each word is stored in 8 consecutive locations, one location each for the corresponding ASCII code, and one location for a null terminator (x00). For example the word "Compute" would be stored as:

x43
x6F
x6D
x70
x75
x74
x65
x00

PROBLEM CONTINUTED ON NEXT PAGE

Name: Jon Dahm

Part a (6 points): Suppose we were to load the contents of location x8000. After the load completes, A, B, C, and D are bytes of data in the row buffer. What are the addresses of the locations containing those bytes of data?



Address containing A:	x8009
Address containing B:	x8001
Address containing C:	x8008
Address containing D:	x8000

Part b (19 points): We wish to determine how many of the 64 English words end in the letter 'e' with a minimum number of memory accesses.

Part b1 (7 points): If we end up having to load 8 rows into the row buffer, how many bits must have been used to specify the row, and how many bits must have been used to specify the column?

$64 \cdot 8 = x40 \cdot x8$

$= x200$

$= 0010\ 0000\ 0000$ (row | col | bank | b-o-b)

Row bits: 5 bits

Column bits: 3 bits

$10 - 3 + 1 = 8$ bits = row + col

Part b2 (6 points): How many clock cycles are necessary to access this information from memory, assuming memory accesses are sent to DRAM in order of increasing addresses? (Note: Your answer will depend on how much you can use interleaving.)

	row	col	bank	bob
x8006 →	...	0000	011	0
x800E →	...	0000	111	0
x8016 →	...	0001	011	0
x801E →	...	0001	111	0

same bank throughout accesses.

$(13 + 3 \cdot 2^3) \cdot 8 = (13 + 24) \cdot 8 = 37 \cdot 8 = 240 + 56 = 296$

(row access time, col access time, num cols accessed, num rows accessed)

Clock cycles: 296

Part b3 (6 points): Suppose we interchange the column bits and the bank bits. Now how many clock cycles are necessary to access this information from memory, assuming memory accesses are sent to DRAM in order of increasing addresses?

	row	bank	col	bob
x8006 →	...	0000	011	0
x800E		0000	111	0
x8016 →		0001	011	0
x801E		0001	111	0

note that you have to load a new row here.

$(13 + 3 \cdot 2) \cdot 8 \cdot 2^2 - (3-1) \cdot (8 \cdot 2^2 - 1)$

(row access time, col access time, num cols accessed, num rows accessed, num banks; overlapping time saved by interleaving)

Clock cycles: 546

$= 19 \cdot 32 - 2 \cdot 31$

$= 19 \cdot 32 - 2 \cdot 32 + 2$

$= 17 \cdot 32 + 2$

$= 320 + 210 + 14 + 2 = 530 + 16 = 546$