First:_____ Middle Initial: _____ Last:_____

This is a closed book exam. You must put your answers on this piece of paper only. You have 50 minutes, so allocate your time accordingly. ***Please read the entire quiz before starting.***

(5) Question 1. Number of decimal digits	
(5) Question 2. Decimal value	
(2) Part 3a. Specify `RegB`	
(2) Part 3b. Specify `0 or 1`	
(1) Part 3c. Specify `0 or 1`	
(5) Question 4. Choose A-E	
(5) Question 5. Show the machine code	
(5) Question 6. Show assembly	

(5) Question 7. Simplified memory cycles (you may or may not need all 5 entries)

R/W	Addr	Data

(5) Question 8. Choose A-G	
(5) Question 9. Give value of number	
(5) Question 10. Effective address	
(5) Question 11. Give op code	
(5) Question 12. Number of bytes/sec	
(5) Question13. Choose A, B, C or D.	
(5) Question 14. Matrix equation	
(5) Question 15. ADC conversion	
(5) Question 16. LED equation	
(5) Question 17. List all registers	
(5) Question 18. Choose A-E	
(5) Question 19a) Choose A-F	
(5) Question 19b) Choose A-F	

(5) Question 1. The measurement system range is 0 to 1999. How many decimal digits is it?

(5) Question 2. What is the unsigned decimal equivalent of the 8-bit hexadecimal $A5?

(5) Question 3. Consider the result of executing the following two 6812 assembly instructions.

```
ldab #100
addb #200
```

(2) Part a) What is the value in Register B after these two instructions are executed? Give your answer in unsigned decimal (0 to 255).
(2) Part b) What will be the value of the carry (C) bit?
(1) Part c) What will be the value of the overflow (V) bit?

(5) Question 4. Which instruction reads the 16-bit TCNT register?

A `ldaa TCNT`
B. `ldab TCNT`
C. `std TCNT`
D. `read TCNT`
E. `ldy TCNT`

(5) Question 5. Show the machine code for the following the instruction

```
ldy 2,sp+
```

(5) Question 6. Write friendly assembly code that makes PORTT bit 6 an output. You may assume the following two definitions.

```
PORTT   equ $00AE
DDRT    equ $00AF
```

(5) Question 7. Give the simplified memory cycles produced when the following one instruction is executed. Assume the PC contains $F000, Register Y contains $0900, each memory location from $0000 to $0BFF contains a value equal to the least significant byte of its address. I.e., $0000 contains $00, $0001 contains $01, etc. Just show R/W=Read or Write, Address, and Data for each cycle.

```
$F000 DD00   ldy 0
```

(5) Question 8. What is the effect of executing these two instructions?

```
pshd
pulx
```

Choose one of the following:

A) "Read from unimplemented I/O port"
B) "Read from uninitialized RAM address"
C) "Read from unprogrammed ROM address"
D) "Assembly syntax error"
E) Value of Reg D is copied to Reg X
F) Value of Reg X is copied to Reg D
G) Values of Reg D and Reg X are exchanged

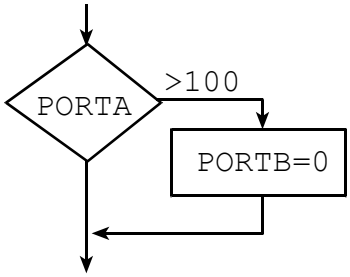
(5) Question 9. A signed 16-bit binary fixed point number system has a Δ resolution of 1/4. What is the corresponding value of the number if the integer part stored in memory is 2000?

(5) Question 10. Assume the PC contains $F000, Register A contains 5, Register X contains $0800, Register Y contains $0900.

```
$F000 EDE4    ldy A,X
```

What is the effective address of this instruction?

(5) Question 11. Assume **PORTA** is an unsigned input and **PORTB** is an output. The goal of this program is to clear **PORTB** if **PORTA** is larger than 100. Which op code should be used in the **???** position?

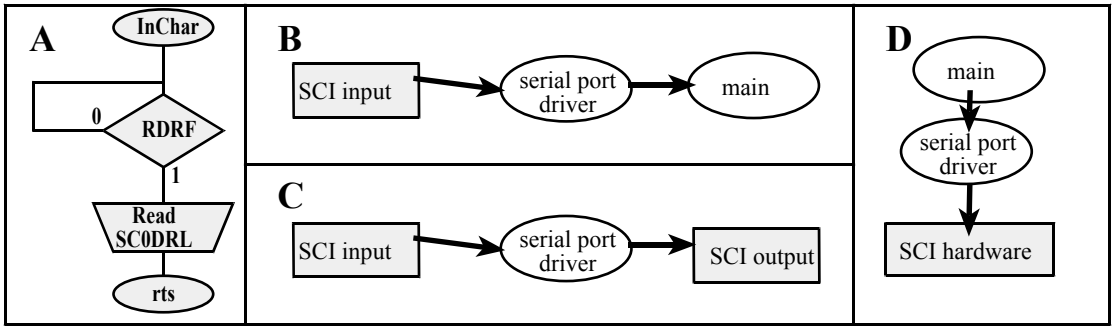



```
        ldaa PORTA
        cmpa #100
        ???  skip
        clr  PORTB
skip
```

(5) Question 12. What is the bandwidth in bytes/sec for a serial channel operating at a baud rate of 9600 bits/sec? There is no parity and one stop bit.

(5) Question 13. Which answer is the **data flow graph** for the following program? The **main** program calls **InChar**.

```
RDRF   equ   $20
InChar brclr SC0SR1,#RDRF,*
       ldaa  SC0DRL          read ASCII character
       rts
```

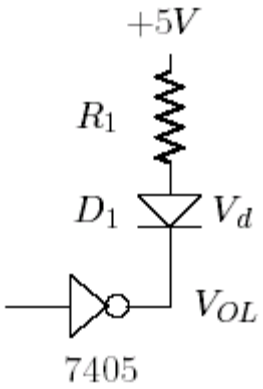


(5) Question 14. Consider a matrix with 4 rows and 7 columns, stored in column-major zero-index format. Each element is 1 byte or 8 bits. Which equation correctly calculates the address of the element at row I and column J?

A. base+I+J
B. base+4*I+J
C. base+I+4*J
D. base+7*I+J
E. base+I+7*J

(5) Question 15. What digital result occurs when the 6812 ADC converts 1.00V?

(5) Question 16. Give the general equation showing LED current I_d as a function of LED voltage V_d, gate output voltage V_{OL}, and resistance R_1.



(5) Question 17. Which registers are pushed on the stack by **`swi`** and pulled off by **`rti`**?

A. all registers but the SP
B. PC
C. PC and CCR
D. all registers including the SP
E. SP

(5) Question 18. Five interpreters were presented in Tutorial 10. Assuming each interpreter was modified to accept 26 commands, labeled A-Z, which technique will have the fastest lookup speed?

A. **direct coding** using the switch statement
B. **array** containing the list of functions to execute
C. **table** containing the letter command and the corresponding function to execute
D. **linked list** containing the letter command and the corresponding function to execute
E. **binary tree** containing the letter command and the corresponding function to execute

(10) Question 19. Consider the following linked list FSM

```
*****A) place it here*******
PORTA equ  0
DDRA  equ  2
*****B) place it here*******
      org  $0800
*****C) place it here*******
      org  $F000
*****D) place it here*******
main  movw #SA,pt
loop  ldx  pt
      movb 0,x,PORTA
*****E) place it here*******
      ldx  1,x
      stx  pt
      bra  loop
      org  $FFFE
      fdb  main
*****F) place it here*******
```

Part a) Where should you place the variable, **`pt`**? Answer A-F.

```
pt    rmb  2
```

Part b) Where should you place the FDM data structure? Answer A-F.

```
SA    fcb  10    output
      fdb  SB    next state
SB    fcb  25    output
      fdb  SA    next state
```

These two tables interpret indexed-mode machine codes

rr	register
00	X
01	Y
10	SP
11	PC

postbyte,xb	syntax	mode	explanations
rr000000	,r	IDX	5-bit constant, n=0
rr00nnnn	n,r	IDX	5-bit constant, n=0 to +15
rr01nnnn	-n,r	IDX	5-bit constant, n=-16 to -1
rr100nnn	n,+r	IDX	pre-increment, n=1 to 8
rr101nnn	n,-r	IDX	pre-decrement, n=1 to 8
rr110nnn	n,r+	IDX	post-increment, n=1 to 8
rr111nnn	n,r-	IDX	post-decrement, n=1 to 8
111rr100	A,r	IDX	Reg A accumulator offset
111rr101	B,r	IDX	Reg B accumulator offset
111rr110	D,r	IDX	Reg D accumulator offset
111rr000 ff	n,r	IDX1	9-bit cons, n 16 to 255
111rr001 ff	-n,r	IDX1	9-bit const, n -256 to -16
111rr010 eeff	n,r	IDX2	16-bit const, any 16-bit n
111rr111	[D,r]	[D,IDX]	Reg D offset, indirect
111rr011 eeff	[n,r]	[IDX2]	16-bit constant, indirect

LDY — Load Index Register Y — LDY

Operation: (M : M + 1) ⇒ Y

Description: Loads the most significant byte of index register Y with the content of memory location M, and loads the least significant byte of Y with the content of the next memory location at M + 1.

S	X	H	I	N	Z	V	C
–	–	–	–	Δ	Δ	0	–

N: Set if MSB of result is set; cleared otherwise.

Z: Set if result is $0000; cleared otherwise.

V: 0; Cleared.

Source Form	Address Mode	Object Code	Cycles	Access Detail
LDY *#opr16i*	IMM	CD jj kk	2	OP
LDY *opr8a*	DIR	DD dd	3	RfP
LDY *opr16a*	EXT	FD hh ll	3	ROP
LDY *oprx0_xysp*	IDX	ED xb	3	RfP
LDY *oprx9,xysp*	IDX1	ED xb ff	3	RPO
LDY *oprx16,xysp*	IDX2	ED xb ee ff	4	fRPP
LDY [D,*xysp*]	[D,IDX]	ED xb	6	fIfRfP
LDY [*oprx16,xysp*]	[IDX2]	ED xb ee ff	6	fIPRfP

```
aba    8-bit add RegA=RegA+RegB
abx    unsigned add RegX=RegX+RegB
aby    unsigned add RegY=RegY+RegB
adca   8-bit add with carry to RegA
adcb   8-bit add with carry to RegB
adda   8-bit add to RegA
addb   8-bit add to RegB
addd   16-bit add to RegD
anda   8-bit logical and to RegA
andb   8-bit logical and to RegB
andcc  8-bit logical and to RegCC
asl/lsl   8-bit left shift Memory
asla/lsla 8-bit left shift RegA
aslb/lslb 8-bit arith left shift RegB
asld/lsld 16-bit left shift RegD
asr    8-bit arith right shift Memory
asra   8-bit arith right shift
asrb   8-bit arith right shift to RegB
bcc    branch if carry clear
bclr   clear bits in memory
bcs    branch if carry set
beq    branch if result is zero (Z=1)
bge    branch if signed ≥
bgnd   enter background debug mode
bgt    branch if signed >
bhi    branch if unsigned >
bhs    branch if unsigned ≥
bita   8-bit and with RegA, sets CCR
bitb   8-bit and with RegB, sets CCR
ble    branch if signed ≤
blo    branch if unsigned <
bls    branch if unsigned ≤
blt    branch if signed <
bmi    branch if result is negative (N=1)
bne    branch if result is nonzero (Z=0)
bpl    branch if result is positive (N=0)
bra    branch always
brclr  branch if bits are clear,
brn    branch never
brset  branch if bits are set
bset   set bits in memory
bsr    branch to subroutine
bvc    branch if overflow clear
bvs    branch if overflow set
call   subroutine in expanded memory
cba    8-bit compare RegA with RegB
clc    clear carry bit, C=0
cli    clear I=0, enable interrupts
clr    8-bit Memory clear
clra   RegA clear
clrb   RegB clear
clv    clear overflow bit, V=0
cmpa   8-bit compare RegA with memory
cmpb   8-bit compare RegB with memory
com    8-bit logical complement to Memory
coma   8-bit logical complement to RegA
comb   8-bit logical complement to RegB
cpd    16-bit compare RegD with memory
cpx    16-bit compare RegX with memory
cpy    16-bit compare RegY with memory
daa    8-bit decimal adjust accumulator
dbeq   decrement and branch if result=0
dbne   decrement and branch if result≠0
dec    8-bit decrement memory
deca   8-bit decrement RegA
decb   8-bit decrement RegB
des    16-bit decrement RegSP
dex    16-bit decrement RegX
dey    16-bit decrement RegY
ediv   RegY=(Y:D)/RegX, unsigned divide
edivs  RegY=(Y:D)/RegX, signed divide
emacs  16 by 16 signed mult, 32-bit add
emaxd  16-bit unsigned maximum in RegD
emaxm  16-bit unsigned maximum in memory
emind  16-bit unsigned minimum in RegD
eminm  16-bit unsigned minimum in memory
emul   RegY:D=RegY*RegD unsigned mult
emuls  RegY:D=RegY*RegD signed mult
eora   8-bit logical exclusive or to RegA
eorb   8-bit logical exclusive or to RegB
etbl   16-bit look up and interpolation
exg    exchange register contents
fdiv   16-bit unsigned fractional divide
ibeq   increment and branch if result=0
ibne   increment and branch if result≠0
idiv   16-bit unsigned divide
idivs  16-bit by 16-bit signed divide
inc    8-bit increment memory
inca   8-bit increment RegA
incb   8-bit increment RegB
ins    16-bit increment RegSP
inx    16-bit increment RegX
iny    16-bit increment RegY
jmp    jump always
jsr    jump to subroutine
lbcc   long branch if carry clear
lbcs   long branch if carry set
lbeq   long branch if result is zero
lbge   long branch if signed ≥
lbgt   long branch if signed >
lbhi   long branch if unsigned >
lbhs   long branch if unsigned ≥
lble   long branch if signed ≤
lblo   long branch if unsigned <
lbls   long branch if unsigned ≤
lblt   long branch if signed <
lbmi   long branch if result is negative
lbne   long branch if result is nonzero
lbpl   long branch if result is positive
lbra   long branch always
lbrn   long branch never
lbvc   long branch if overflow clear
lbvs   long branch if overflow set
ldaa   8-bit load memory into RegA
ldab   8-bit load memory into RegB
ldd    16-bit load memory into RegD
lds    16-bit load memory into RegSP
ldx    16-bit load memory into RegX
ldy    16-bit load memory into RegY
leas   16-bit load effective addr to SP
leax   16-bit load effective addr to X
leay   16-bit load effective addr to Y
lsr    8-bit logical right shift memory
lsra   8-bit logical right shift RegA
lsrb   8-bit logical right shift RegB
lsrd   16-bit logical right shift RegD
maxa   8-bit unsigned maximum in RegA
maxm   8-bit unsigned maximum in memory
mem    determine the membership grade
mina   8-bit unsigned minimum in RegA
minm   8-bit unsigned minimum in memory
movb   8-bit move memory to memory
movw   16-bit move memory to memory
mul    RegD=RegA*RegB
neg    8-bit 2's complement negate memory
nega   8-bit 2's complement negate RegA
negb   8-bit 2's complement negate RegB
oraa   8-bit logical or to RegA
orab   8-bit logical or to RegB
orcc   8-bit logical or to RegCC
psha   push 8-bit RegA onto stack
pshb   push 8-bit RegB onto stack
pshc   push 8-bit RegCC onto stack
pshd   push 16-bit RegD onto stack
pshx   push 16-bit RegX onto stack
pshy   push 16-bit RegY onto stack
pula   pop 8 bits off stack into RegA
pulb   pop 8 bits off stack into RegB
pulc   pop 8 bits off stack into RegCC
puld   pop 16 bits off stack into RegD
```

```
pulx   pop 16 bits off stack into RegX
puly   pop 16 bits off stack into RegY
rev    Fuzzy logic rule evaluation
revw   weighted Fuzzy rule evaluation
rol    8-bit roll shift left Memory
rola   8-bit roll shift left RegA
rolb   8-bit roll shift left RegB
ror    8-bit roll shift right Memory
rora   8-bit roll shift right RegA
rorb   8-bit roll shift right RegB
rtc    return sub in expanded memory
rti    return from interrupt
rts    return from subroutine
sba    8-bit subtract RegA=RegA-RegB
sbca   8-bit sub with carry from RegA
sbcb   8-bit sub with carry from RegB
sec    set carry bit, C=1
sei    set I=1, disable interrupts
sev    set overflow bit, V=1
sex    sign extend 8-bit to 16-bit reg
staa   8-bit store memory from RegA
stab   8-bit store memory from RegB
std    16-bit store memory from RegD
sts    16-bit store memory from SP
stx    16-bit store memory from RegX
sty    16-bit store memory from RegY
suba   8-bit sub from RegA
subb   8-bit sub from RegB
subd   16-bit sub from RegD
swi    software interrupt, trap
tab    transfer A to B
tap    transfer A to CC
tba    transfer B to A
tbeq   test and branch if result=0
tbl    8-bit look up and interpolation
tbne   test and branch if result≠0
tfr    transfer register to register
tpa    transfer CC to A
trap   illegal op code, or software trap
tst    8-bit compare memory with zero
tsta   8-bit compare RegA with zero
tstb   8-bit compare RegB with zero
tsx    transfer S+1 to X
tsy    transfer S+1 to Y
txs    transfer X-1 to S
tys    transfer Y-1 to S
wai    wait for interrupt
wav    weighted Fuzzy logic average
xgdx   exchange RegD with RegX
xgdy   exchange RegD with RegY
```

example	addressing mode	Effective Address
ldaa #u	immediate	none
ldaa u	direct	EA is 8-bit address (0 to 255)
ldaa U	extended	EA is a 16-bit address
ldaa m,r	5-bit index	EA=r+m (-16 to 15)
ldaa v,+r	pre-increment	r=r+v, EA=r (1 to 8)
ldaa v,-r	pre-decrement	r=r-v, EA=r (1 to 8)
ldaa v,r+	post-increment	EA=r, r=r+v (1 to 8)
ldaa v,r-	post-decrement	EA=r, r=r-v (1 to 8)
ldaa A,r	Reg A offset	EA=r+A, zero padded
ldaa B,r	Reg B offset	EA=r+B, zero padded
ldaa D,r	Reg D offset	EA=r+D
ldaa q,r	9-bit index	EA=r+q (-256 to 255)
ldaa W,r	16-bit index	EA=r+W (-32768 to 65535)
ldaa [D,r]	D indirect	EA={r+D}
ldaa [W,r]	indirect	EA={r+W} (-32768 to 65535)

Motorola 6812 addressing modes