

ECE 341F - Computer Organization

Department of Electrical and Computer Engineering

November 1999

Mid-term Examination (Weight = 20%)

Answer all questions. Write your answers on the exam paper.

Each question has a different assigned value, as indicated.

Open textbook and open lab manual; no written notes and no written or extra printed lab material

Total marks available are 80 marks.

Verify that you exam has all 11 pages.

Name	Solution Set
Student ID.	

Lab Day (circle one): Tues Wed

Question 1: /14

Question 2: /16

Question 3: /10

Question 4: /20

Question 5: /20

Total: /80

Question 1 (14 marks; 2 marks each). *Short Answer Questions.*

a) Name two functions that the Status Register (SR) performs in the 68000 processor.

1. Holds condition codes

2. Holds cpu priority

b) Why does the following instruction give an error during assembly?

`cmpi.b d0,#10`

The operands are in the wrong order.

c) When an interrupt request (say level 4) is made to the 68000, this causes the CPU to transfer control to the Interrupt Service Routine (ISR) specified by the interrupt-vector address for that priority level. As soon as the CPU starts executing the ISR, the interrupt line is still high. What prevents the CPU from branching back immediately to the start of the ISR (because it notices the interrupt)?

The interrupt priority level is set to 4 in the status register, preventing a new level 4 interrupt.

d) The 68000 cannot respond to one interrupt while servicing another. True or false?

False

e) Explain the two concepts polling and memory mapped I/O in the space provided (in point form).

Polling: A loop which executes until a polled device has asserted a bit in a status register indicating readiness of the device.

Memory-Mapped I/O: An I/O device which is accessed through addresses on the bus, as if it was a memory address.

f) A subroutine ADD(X,Y), which has two 32-bit integer parameters X and Y, is called using the stack on the 68000. The subroutine parameters are long words and are pushed on the stack from left to right (i.e. first X, then Y). Registers D0, A0, and A4 must be saved. Register A6 is used as the stack pointer. What is the offset (from (A6)) necessary to access parameter X? What is the offset needed to access D0?

(Assume callee save, if you need to).

Assume D0, A0 and A4 saved using movem instruction

- Different answers for D0 as L or W, B

A6 ->	A4
	A0
	D0
	PC+2
	Y
	X

	L	W,B
X	20	18
D0	8	8

g) A device that interrupts the 68000 at level 3 in the auto-vector mode will go to what address in the Interrupt Vector Table? (Give the address in hex.)

\$6c

Question 2 (16 marks). *Instructions and their operation.*

(a) Assume a0 contains \$12000, what memory address does the following instruction access? (Answer in hex) (2 marks)

```
move.l #9, $8FF8(a0)
```

```
$1aff8
```

(b) What syntax error(s) are there in the following code fragment? What values reside in a0 and d0 after the execution of the following code fragment? (assuming of course that any syntax errors are fixed properly!) (4 marks)

```
move.l #12345678,a0 ® movea.l #12345678, a0 (2 marks)
```

```
move.l #87654321,d0
```

```
move.w d0, -(a0)
```

```
a0: $12345676 (1 mark)
```

```
d0: $87654321 (1 mark)
```

(c) What are the contents of a0 and memory locations \$30000 to \$30006 at the end of the execution of the following code fragment.

```
org $30000 (5 marks)
```

```
dc.w $1234,$5678,$CCDD,$EEFF
```

```
org $20000
```

```
movea.l #30002,a0
```

move.w (a0)+, (a0)+

30000	\$1234
30002	\$5678
30004	\$5678
30006	\$ EEFF
a0	\$30006

(d) Before the execution of the code fragment below, register d5 contains the value 0. What is the value of d5 after the two instructions below have been completed?
\$AAB

(2 marks)

movea.l #LIST, a0

move.w 2(a0), d5

LIST dc.w \$100

dc.w \$aabb

dc.w \$1234

(e) How many memory accesses are required to fetch and execute the following instructions?
(3 marks)

rte 4

dbra d0, WAIT 2

add.l d0, (a0) 5

Question 3 (10 marks). Processor Interface.

Consider the keyboard interface given below. How long after the processor asserts its Ready signal does it see a stable Accept signal?

Allow 3nsec for bus delay and skew. Assume the delay through a gate is 500psec, the time to turn off a tri-state gate is 1nsec, and that the address decoder is a two-level combinational logic circuit.

Also estimate the maximum length of the bus (in centimeters) for this system.

State any assumptions that you make.

Calculations for Question 3:

Assumptions: (1 mark)

- $R/\overline{\text{R}}$, address (incl. A0) are stable at inputs to AND gates when Ready is asserted at input to circuit.

Notes:

- Status flag doesn't affect Accept signal.
- System always runs slower than speed of light.
- System bus includes data, address and control.

Delay = bus delay + AND delay + OR delay + tristate delay + bus delay

$$= 3\text{ns} + 0.5\text{ns} + 0.5\text{ns} + 1\text{ns} + 3\text{ns} = 8\text{ns} \text{ (5 marks)}$$

length = delay x speed

$$= 3\text{ns} \times \boxed{\text{X}} = 90 \text{ cm}$$

bus < 90 cm (4 marks)

Question 4 (20 marks). 68000 Assembly Language Programming.

The program below is supposed to continuously poll the keyboard. Whenever a key is pressed, the program is supposed to print the character to the screen 10 times, and then print a carriage return. However, as it is written this program contains several logical and syntactical errors. You are to correct each of the incorrect instructions on the line adjacent to that instruction. Do not "correct" any instructions that do not have mistakes in them or you will be penalized. You do not have to delete any instructions to make this program work; you only have to correct the flaws in the existing instructions.

SRA equ \$FFFFFF7E3

RBA equ \$FFFFFF7E7 The addresses should be changed to Port B addresses

TBA equ \$FFFFFF7E9

org \$20000

OLOP bsr READ _____

clr.b d1 _____

ILOOP bra PRINT I LOOP bsr PRINT

Add.b 1,d1 add.b #1, d1

cmp.l #10,d1 cmp.b #10, d1

beq ILOOP bne ILOOP

move.b #CR, d0 _____

bra PRINT bsr PRINT

bra ILOOP bra OLOP

*** The READ subroutine is supposed to read a character from the keyboard via polling.**

READ btst.b 0, SRA READ btst.b #0, SRA

beq READ _____

move.w RBA,d0 move.b RBA, d0

rts _____

*** The PRINT subroutine is supposed to send a character to the screen via polling.**

PRINT btst.b #1, SRA PRINT btst.b #2, SRA

beq PRINT _____

move.b d0,#TBA move.b d0, TBA

rts _____

-2 if not found

-1 if found but not corrected properly

-1 if corrected instruction with no mistake

Question 5 (20 marks). Subroutines and Stacks.

You are to write a 68000 assembler language subroutine merge that takes two arrays of numbers (16-bit words), assumed to be sorted in increasing order, and merges them so that the combined array is also sorted. The two arrays are passed to the subroutine by reference, as four arguments on the stack. The merged array is placed in a memory location pointed to by a fifth argument also passed on the stack. The subroutine assumes that memory corresponding to the size of the combined array is correctly allocated at this address. It also assumes that the space allocated for the three arrays do not overlap in memory.

A fragment of the main program that calls merge is shown below, to specify the order and sizes of the arguments on the stack. Your subroutine must conform to this order and these sizes. The subroutine must save the values of all registers that it affects, and restore them before returning.

The fragment of the main program that calls your program is shown below:

l1 move.l a2,-(sp) ; target address for merged array

move.w d1,-(sp) ; size of the second array

move.l a1,-(sp) ; start address of the second array

move.w d0,-(sp) ; size of the first array

move.l a0,-(sp) ; start address of the first array

bsr merge ; call your subroutine

a) Assume that before I1 is executed the stack pointer value is \$30020. What is the value of the stack pointer immediately after the bsr instruction is executed? Fill in the table below showing the address and value of every item that has been put on the stack immediately after the bsr merge instruction has executed. Each line in the table below is two bytes.

Stack: (6 marks)

Address Value

3000C	PC + 2
30010	a0
30014	d0
30016	a1
3001a	d1
3001C	a2

30020	

-2 decimal
-2 increasing stack
1 mark each address-content pair

b) Write the subroutine merge described above. Precede every section of your program with detailed comments explaining what you are trying to accomplish. Marks will be deducted for improper or inadequate comments. Attempt to make your program as short and simple as possible, since marks will be deducted for unnecessarily complex and inefficient programs.

(14 marks)

save registers 1

restore registers 1

boundary conditions 5

proper merging 5

rts 1

style 1

Code for Question 5

```
* save all registers

* assert the size of first array - sizeA is in d0

* assert the size of second array - sizeB is in d1

* assert start address of the first array - addressA is in a0

* assert start address of the second array - addressB is in a1

* assert start address of the third array - addressC is in a2

* loop

* if (sizeA == 0) then -

* while (sizeB != 0) do |

* *addressC = *addressB |

* addressB = addressB + 1 |

* addressC = addressC + 1 | - section A

* sizeB = sizeB - 1 |

* end while |

* exit the program |

* end if _

* if (sizeB == 0) thena -

* while (sizeA != 0) do |

* *addressC = *addressA |

* addressA = addressA + 1 |

* addressC = addressC + 1 | - section B

* sizeA = sizeA - 1 |

* end while |
```

```
* exit the program |
* end if _
* if *addressA <= *addressB then -
* *addressC = *address A | - section C
* sizeA = sizeA - 1 |
* end if _
* else -
* *addressC = *addressB | - section D
* sizeB = sizeB - 1 |
* end if -
* end loop
* check the size of the second array
org $20000
array1 dc.w 1,3,5,8,9
org $21000
array2 dc.w 2,4,6,7,10
org $22000
array3 ds.w 10
org $30000
move.l #array1,a0
move.l #array2,a1
move.l #array3,a2
move.w #5,d0
```

```
move.w #5,d1
l1 move.l a2,-(sp)
move.w d1,-(sp)
move.l a1,-(sp)
move.w d0,-(sp)
move.l a0,-(sp)
bsr merge
trap #15
merge movem.l d0-d7,-(sp)
movem.l a0-a6,-(sp)
*****
* Section A *
*****
secA cmpi.w #0,d0
bgt secB
secALoop cmpi.w #0,d1
ble secAEnd
move.w (a1)+,(a2)+
subi.w #1,d1
bra secALoop
secAEnd bra End
*****
* Section B *
```

secB cmpi.w #0,d1

bgt secC

secBLoop cmpi.w #0,d0

ble secBEnd

move.w (a0)+,(a2)+

subi.w #1,d0

bra secBLoop

secBEnd bra End

*** Section C ***

secC move.w (a0),d4

move.w (a1),d5

cmp.w d4,d5

ble secD

move.w (a0)+,(a2)+

subi.w #1,d0

bra endLoop

*** Section D ***

secD move.w (a1)+,(a2)+

subi.w #1,d1

endLoop bra secA

End movem.l (sp)+,d0-d7

movem.l (sp)+,a0-a6

rts