

## ECE341F -- Computer Organization

Midterm, November 2000 -- Answers

### Question 1 (15 marks)

**1a.** Perform the operations indicated below, showing both the results and the resulting condition codes N, Z, V and C (as in the 68000). All the numbers are 6-bit signed integers in 2's complement representation.

- i. [4 marks]  $011101 + 100011 = (1) 000 000$  N=0 Z=1 V=0 C=1
- ii. [4 marks]  $010101 + 011010 = 101 111$  N=1 Z=0 V=1 C=0

**1b.** A longword object is stored at location 100 as follows:

```
org 100
M      dc.b 'CAT'
```

In addition, longword objects are being kept in a user-defined stack, with the top of the stack pointed to by register A3.

i. [1 mark] Write the instruction(s) to push an item that is stored at the memory location M onto the stack.

```
move.l M, -
(a3)
```

ii. [1 mark] Write the values in hexadecimal (i.e., base 16) describing the contents of the longword at the top of the stack.

```
C A T
undefined byte
43 41 54 xx
```

iii. [1 mark] Write the instruction(s) to pop an item off the stack and store it at memory location N.

```
move.l (a3)+,
N
```

**1c.** The following instructions are intended to cause the program to branch to the label NEXT if the value at address 8(a7) is less than the contents of register d5:

```
cmp d5, 8(a7)
blt NEXT
```

[2 marks] However, these instructions will cause an error to occur when they are assembled. Why does the error occur? Assume that the label NEXT is properly defined elsewhere in the program.

*As per Appendix C, the cmp instruction does not allow d(An) as an addressing mode for the destination. Must use d = Dn and s = d(An) instead.*

[2 marks] Change the instructions to remove the error. Do not add any more instructions or directives; change only what is there already. Be sure that the instructions still perform the conditional branch as described above.

```
cmp 8(a7), d5
bge NEXT
```

### Question 2 (10 marks)

The following Motorola 68000 assembly program fragment takes a 16-bit number as input in register d0, and produces an 8-bit output in memory location **Result**, based on some property of the input number. The listing below also provides, in the left-most column, the addresses of the instructions and data.

```
                                org      $20000
020000:  start      clr.l    d2
020002:                                move.w  d0,d2
020004:  Q2       divu    #7,d2
020008:                                swap.w  d2
02000a:                                move.b  #1,Result
020012:                                tst.w   d2
200014:  Q4       beq     yes
200018:                                clr.b  Result
20001e:  yes      trap    #15
200020:  Result   ds.b   1
```

Answer the following questions concerning this program.

**2a. [1 mark]** What is the value of the **Program Counter (PC)**, in base 16, after the instruction labelled **Start** in the program has been fetched and executed, but before the subsequent instruction has been executed?

$PC = \$20002$

**2b. [2 marks]** If the contents of register d0 is the number  $29_{10}$  before this program begins, show the contents of register **d2**, in base 16 (hexidecimal) **after** the instruction labelled **Q2** has been executed.

$d2 = \$00010004$

(note that *divu* puts the quotient in the lower word of the destination, and the remainder in the upper word.)

**2c. [6 marks]** What does this program do? (In one sentence, describe the function of this program).

*Result is set to 1 (true) if the number in d0 is divisible by 7, i.e., the division leaves no remainder, otherwise it is set to 0.*

**2d. [1 mark]** If the contents of register d0 is the number  $14_{10}$  before this program begins, what is the contents of the program counter, in base 16, after the instruction labelled **Q4** has been executed, but before another instruction is executed?

$PC = \$2001e$

### Question 3 (15 marks)

**[10 marks for Program, 5 marks for Comments]** Write a Motorola 68000 Assembly Language program that determines the lowest bit number in a long word that is a '1'. For example, the following long word has its first '1' bit in bit position number 7 (recall that bits are numbered from 0, from the right, least significant bit):

```
01000010100010010101010010000000
  ^                               ^   ^
  ||                               ||   ||
bit 31                           bit 7  bit 0
```

Assume the following:

- That the long word to be tested is in register d0.
- The answer (i.e., the bit number) should be placed into register d1.
- If the word is equal to zero, then return the number 32.

**Be sure to include comments that describe the function of the program.**

```
        clr.b d1          ; Set bit to test to 0
initially
CHECK   btst.l d1, d0     ; test d1th bit of d0
        bne RETURN       ; if one, branch to
end, otherwise
        addq.b #1,d1      ; increment d1 to check
next bit
```

```

        cmpi.b #32,d1    ; see if we've checked
all the bits
        bne CHECK      ; if not, continue
checking
RETURN trap #15        ; otherwise, end

```

#### Question 4 (15 marks)

Consider the following program:

Assume interrupts have been properly set-up and initialized on the 68K UltraGizmo. An interrupt request occurs (and is accepted) while the processor is executing the `addi` instruction indicated below.

```

                                stack    equ $40000
                                org      $20000
$20000                          movea.l #stack,a7
20006                          bsr     sub1
2000a                          trap    #15

                                org      $30000
$30000                          sub1   clr.l d0
30002                          addi.b #1,d0
30006                          move.w d0,-(a7)
30008                          bsr     sub2
3000a                          move.w (a7)+,d0
3000c                          rts
3000e                          sub2   clr.l d1
30010                          addi.b #1,d1
30014                          addi.b #1,d1
30018                          move.b d1,d2
3001a                          rts
3001c                          inter  cmp.b d1,d2
3001e                          bne    sub1
30020                          rte

```

**4a. [5 marks]** In the boxes on the left hand side, write the address location of each instruction. (*Shown in italics*).

**4b. [1 mark]** One instruction has a serious error. Identify the instruction and explain the problem.

*The instruction `bne sub1` in the interrupt service routine is erroneous -- branching out of the ISR into another subroutine will destroy the integrity of the stack.*

**4c. [9 marks]** Show the contents of the stack after instruction at location **inter** is executed. Note that you must complete part (a) before you can complete part (c).

Stack Address	Stack Contents	
3FFF0	[SR]	
3FFF2	30010	(2 words)
3FFF4	^^	
3FFF6	30008	(2 words)
3FFF8	^^	
3FFFA	1	(contents of d0)
3FFFC	2000A	(2 words)
3FFFE	^^	
40000		(empty)

**Question 5 (20 marks)**

Consider the single bus architecture shown in Figure 1 below. ("acc" is an internal register known as an accumulator). The memory of this computer is word addressable. Assume standard memory control signals R/W and MFC are available.

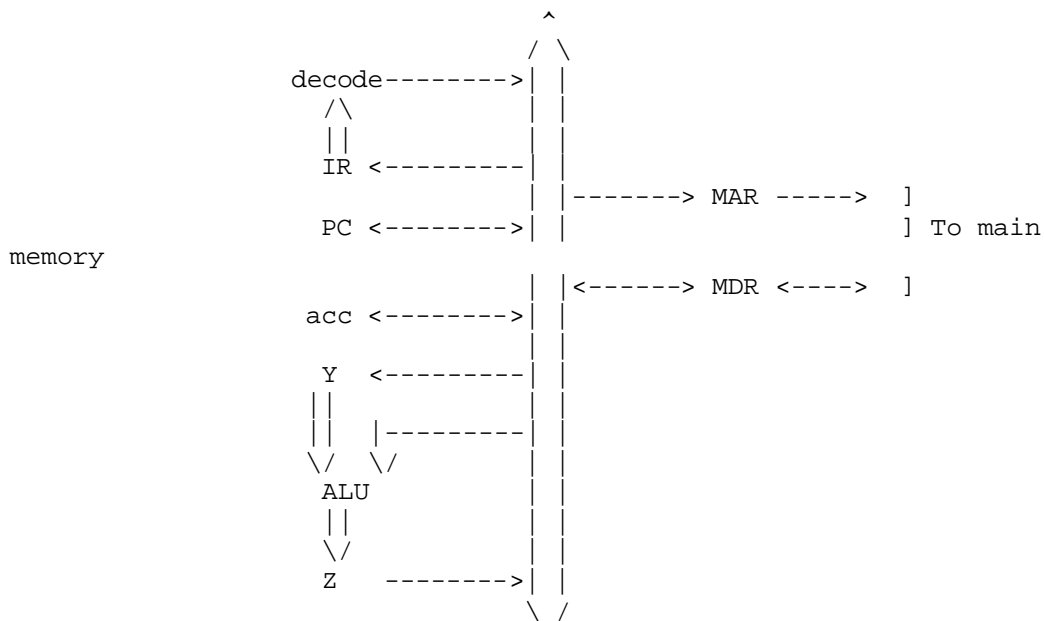


Figure 1

Assume that it has the following instructions:

load X	acc <--- [X]
loadi X	acc <--- [[X]]
loadx X	acc <--- [X + [acc]]

add X            acc <--- [acc] + [X]  
 inc X            X <--- [X] + 1

Assume that the control signal IR\_X\_out will drive the X field of the instruction onto the bus. The clr\_y signal will clear the Y register at the start of the cycle, in time to be used by the ALU in the same cycle.

a) [4 marks] Write an appropriate control step sequence for the fetch cycle. Fetching a single word retrieves both the instruction and the operand X.

*The instruction fetch sequence is the same for all instructions, and looks like this:*

*PC\_out, MAR\_in, READ, clr\_y, carry\_in,  
 ADD, Z\_in  
 Z\_out, PC\_in, WMFC  
 MDR\_out, IR\_in*

b) Write appropriate control step sequences for each of the five instructions above.

load X [2 marks]

*acc <-- [X]      Move the contents of  
 memory location X into the accumulator  
 register.*

*IR\_X\_out, MAR\_in, READ, WMFC  
 MDR\_out, acc\_in*

loadi X [3 marks]

*acc <-- [[X]]    Move the contents of the  
 address located at memory location X into  
 the accumulator register.*

*IR\_X\_out, MAR\_in, READ, WMFC  
 MDR\_out, MAR\_in, READ, WMFC  
 MDR\_out, acc\_in*

loadx X [4 marks]

*acc <-- [X + [acc]]    Add the number X to  
 the contents of the accumulator to obtain an  
 address, and fetch the contents.*

*IR\_X\_out, Y\_in  
acc\_out, ADD, Z\_in  
Z\_out, MAR\_in, READ, WMFC  
MDR\_out, acc\_in*

add X **[4 marks]**

*acc <-- [acc] + [X] Add the contents of the  
accumulator to the contents of memory  
location X*

*IR\_X\_out, MAR\_in, READ  
acc\_out, Y\_in, WMFC  
MDR\_out, ADD, Z\_in  
Z\_out, acc\_in*

inc X **[3 marks]**

*X <-- [X] + 1 Add one to the contents of  
memory location X*

*IR\_X\_out, MAR\_in, READ, WMFC  
clr\_y, carry\_in, MDR\_out, Z\_in, ADD  
Z\_out, MDR\_in, WRITE, WMFC*