150

COMP.3050.201 Test 1 (200 points) 2 hours 3/4/2019

CLOSED BOOK/ LAPTOP. No calculators. One page (both sides) of notes are allowed.

NAME: phong Vo

Give specific numbers where appropriate, not a general verbal description.

1. (10 pts) Convert the base 10 real number 142.875 into

Base 2 (4 pts) 1000 1110.111 ✓

Base 8 (2 pts) 216.7 ✓

Base 16 (4 pts) 8E.E ✓

10

2. (20 pts each) Given the following 32 bit sequence (copied below):

$M = 2^{-1} + 2^{-3} + 2^{-5} + 2^{-6}$

$^{64}\ ^{2}\ ^{-1}\ ^{-3}\ ^{-5}\ ^{-6}$

0 | 1 0 0 0 0 1 0 | 1 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Sign | exponent and mantissa components

A. $S = +$ If the sequence represents a signed magnitude floating point value

$\leftarrow E = 66$ using the IBM format discussed in class, what is the base 10 floating

-64 point value of the sequence?

$= 16^{2} \times \left(2^{-1} + 2^{-3} + 2^{-5} + 2^{-6}\right) = 2^{7} + 2^{5} + 2^{3} + 2^{2}\ (10) = 172_{(10)}$ ✓

$= 2^{8}$ $= 128 + 32 + 8 + 4$

$^{7}\ ^{6}\ ^{5}\ ^{4}\ ^{3}\ ^{2}\ ^{1}\ ^{0}\ ^{-1}\ ^{-2}\ ^{-4}\ ^{-5}$

0 | 1 0 0 0 0 1 1 0 | 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Sign | exponent and mantissa components

$E = 2^{7} + 2^{2} + 2^{1}$

$= (128 + 4 + 2) - 127$

$= 7$

$\Rightarrow E = 2^{7}$

$M =$

B. $S = +$ If the sequence represents a signed magnitude floating point value

$E = 7$ using the IEEE 754 single precision format discussed in class, what is the base 10 floating point value of the sequence?

$= 2^{7} \times \left(1 + 2^{-2} + 2^{-4} + 2^{-5}\right) = 2^{5} + 2^{3} + 2^{2}\ (10) = 44_{(10)}$ ✗

$= 32 + 8 + 4$ -5

Remember to add 1 in IEE 754 format

35

3. The following bits represent an IEEE 754 single precision floating point number:

FLOAT: 0 1 0 0 0 0 1 1 0 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

A. (10 points) Show the bits after the number it represents has been multiplied by the base 10 number 256 $= 2^8$

0 | 1000 1110 | 0110 1000 0 ... ✓

B. (15 pts) If the same floating point number (**before** the multiply) is stored beginning at address 100 on a Big Endian computer, show the number in hexadecimal at the specific memory addresses it uses. **Leave blank any addresses it does not use**. Fill in **ONLY** the addresses that the floating point number uses.

FLOAT: 0 1 0 0 | 0 0 1 1 | 0 0 1 1 | 0 1 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

4 3 3 4

100	43 ✓	105	✓
101	34 ✓	106	✓
102	00 ✓	107	✓
103	00 ✓	108	✓
104	✓	109	✓

25

4. (5 points each)

a) Denormalized floating point numbers allow for what type of values to be represented?
   a. Huge numbers close to infinity
   b. Tiny numbers close to zero ✓
   c. NANs – Not A Numbers

b) The memory address space on the Mic1 is best described as:
   a. Byte-addressable (each byte has its own address), Big Endian
   b. Byte-addressable, Little Endian
   c. 16-bit addressable (every 16 bits has its own address) with 2 to the 16th addresses
   d. 16-bit addressable with 2 to the 12th addresses ✓

c) If a 16-bit signed number is expressed in hex as CE74, what would its sign-extended value be as a 32-bit number?

FFFF CE74 ✓

15

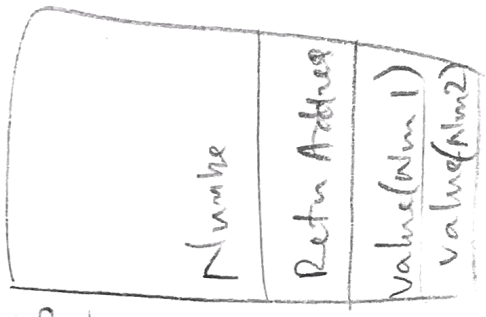
5. (40 pts) You are to write a function Max in Mic-1 assembly language that takes the larger of two numbers **passed by reference** on the stack. That is, the addresses of the two numbers are on the stack, not their values. Do **not** write a function that is based on pass by value. That will not be worth **any** points.

Your function returns the maximum value in AC. Show only the function, **not** the calling code. You **must** provide reasonable comments to aid in grading. Assume that the call to your function is made such that the following values are on the stack:

SP points to the location that holds the return PC
SP +1 points to the address of Num1
SP+2 points to the address of Num2

Max: ←Write the Max function

```
Max:    LODL 1  ; Num2
        SUBL 2  ; Num2 - Num1
        JNEG bigOp1:
        LODL 1
        RETN    ; Num2
bigOp1: LODL 2
        RETN    ; Num1
```




```
Max: LODL 1   ; get address of Num1
     PSHI     ; push value of Num1 in stack
     LODL 3   ; get address of Num2
     PSHI     ; push value of Num2 on stack
     POP      ;
     LODL 0   ; AC = Num2
     SUBL 2   ; subtract so AC = Num2 - Num1
     JNEG Numbigger
     LODL 0   ; get num2, the larger number into the AC
     INSP 2   ; clean up stack for RETN
     RETN     WORKS
Numbigger: LODL 1  ; get Num1, the larger number into the AC
     INSP 2   ; clean up stack
     RETN
```

10

6. (30 pts total) This is an alternate version of the self-modifying code we looked at in class (the code modifies the smc1: instruction). Execution begins at main:. Show on the line below the final values of r0:, r1:, r2:, r3:, and r4:. **No partial credit for incorrect answers.**

```
c1: 1
c5: 5
index: 5
r0: 0
r1: 0
r2: 0
r3: 0
r4: 0
main:  lodd index: 5
       jzer done:
       subd c1: 5-1
       stod index: 4
       lodd c5: 5
       addd c5: 10
       stod c5: 10   c5=10
smc1:  stod r0: 10
       lodd smc1:
       addd c1:
       stod smc1:
       jump main:
done:  halt
```

r0:= 10 r1:= 20 r2:= 40 r3:= 80 r4: 160

30

7. (40 points) The MIC-1 bit format is shown below. You should be familiar with all the fields and how they are used. Also below are 4 MAL instructions. Indicate if a given MAL is valid or invalid for MIC-1, and, if valid, fill in the **DECIMAL** (i.e. bits 1101 are filled as 13) values for each field **in the space provided**. **If invalid, write below the figure why not, but in case you are wrong fill in as many of the fields as you can.**

Register designations are as follows: pc=0 (prog counter) ac=1 (accumulator) sp=2 (stack ptr) ir=3 (instr reg) tir=4 (tmp inst reg) zr=5 (fixed zero) po=6 (plus 1) no=7 (minus 1) amask=8 (addr msk) smask=9 (stack msk) a=10(a scratch) b=11(b scratch) c=12(c scratch) d=13(d scratch) e=14(e scratch) f=15(f scratch)

A. mbr := lshift(band(ir,amask));ir := lshift(band(ir,amask));goto 42;wr;
B. f := mbr + 1; mar := pc;rd;
C. tir :=lshift(band(tir,mar));if n then goto 48;
D. tir :=lshift(band(tir,mbr));if z then goto 77;

VALID?	AMUX	COND	ALU	SH	MBR	MAR	RD	WR	ENC	C	B	A	ADDR	
YES	0	3	1	2		0	0		1	3	8	3	42	
NO		0								15	6			
NO			1	2									48	
YES		1 (2)		2	0	0	0	0		4	4		77	X-1

AMUX	COND	ALU	SH	MBR,MAR,RD,WR,ENC
0=A latch	0 = no jmp	0 = A + B	0 = no shift	0 = no
1=MBR	1 = jmp if n=1	1 = A and B	1 = shift rt	1 = yes
	2 = jmp if z=1	2 = A	2 = shift lt	
	3 = always jmp	3 = not A		

Invalid rows **Why invalid**

B For ~ -7

C ! ~ -7

25