CPSC 240 Spring 2021

Computer Organization & Assembly Language Programming Mock Final

Name: SI Students

|  |  |
| --- | --- |
| Section | Points |
| I. Math Problems [MATH] | /? |
| II. General Knowledge [GK] | /? |
| III. GDB [GDB] | /? |
| IV. Short Programs [SP] | /? |
| Total: | /? |

Instructions:

1. This is an open-book, open-notes, open-internet mock exam.
2. Calculators are allowed but discouraged. We want to practice the fundamentals.
3. Communication with classmates is encouraged; however, do NOT collaborate on the real exam.

Notes:

* Contribute!
  1. Any person with a link can comment on solutions (suggest edits or approve of good submissions)
* Why submit solutions?
  1. By submitting your solutions, you allow the opportunity for others to critique and respond to your answer.

○ You may not be able to solve all the questions on your own (due to lack of time or unfamiliarity on the subject), but others may have t he information for you.

# Math Problems [MATH]

For each question in this section, there is one best p ossible answer. Calculators are allowed but not suggested. Show all work if applicable.

[MATH-01] Convert 42(2/3) to a 64-bit IEEE-754 number. A nswer in Hex.

42 = 101010

2/3 x 2 = 1 + 1/3

1/3 x 2 = 0 + 2/3

2/3 x 2 = 1 + 1/3

101010.1010101010101010101…. x 2 ^0

1.01010101010101010……. x 2 ^ 5 5 = true exp

Bias + true exponent = 1023 + 5 = 1028 = 100 0000 0100 = stored exponent

0100 0000 0100 0101 0101 0101 0101 0101 0101 0101 0101 0101 0101 0101 0101 0101

0x4045 5555 5555 5555

[MATH-02] Convert 52.125 to a 64-bit IEEE-754 number—answer in Hex.

52 = 110100

.125 x 2 = 0 + .250

.250 x 2 = 0 + .50

.5 x 2 = 1 + 0.0

52.125 = 110100.001 x 2 ^ 0

1.10100001 x 2 ^ 5 true exponent = 5

1023 + 5 = 1028 our stored exponent

1028 = 100 0000 0100

All together

0100 0000 0100 1010 0001 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000

0x404A 1000 0000 0000

[MATH-03] Convert − 4371 to an IEEE754 64-bit float.

-43 = 101011

1/7 x 2 = 0 + 2/7

2/7 x 2 = 0 + 4/7

4/7 x 2 = 1 + 1/7

-43 1/7 = 101011.001001001001001001001….. x 2^0

1.0101 1001 0010 0100 1001 0010 0100 1001 001…. x 2 ^5

5 = true exponent

1023 + 5 = 1028 = 100 0000 0100

Put it all together

0100 0000 0100 0101 1001 0010 0100 1001 0010 0100 1001 0010 0100 1001 0010 0100

0x4045 9249 2492 4924

[MATH-04] Convert the float number 0x3FD8 0000 0000 0000 to a decimal.

Binary: 0011 1111 1101 1000 0000 0000 0000 0000 0000 0000 0000 000 0000 0000 0000 0000

Determine its positive = 0 on first bit

011 1111 1101 = 1+4+8+16+32+64+128+256+512 = 1021

1021 – 1023 = -2

1.1000 0000 0000…. X 2 ^ -2

= 11.0000 0000 0000 . x 2^-3

3 x 2^-3 = 3/8 = .375

[MATH-05] Represent the IEEE-754 double 0xBFEF C000 0000 0000 as a decimal.

Binary: 1011 1111 1110 1111 1100 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000

Mantissa = 011 1111 1110 = 2+4+8+16+32+64+128+256+512=1022

1. – 1023 = -1 true exponent

1.1111 1100 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000

1111111.00 0000 0000 00000 x 2 ^-7

1 + 2 + 4 +8 +16 + 32 + 64 = 127 x 2 ^ -7 = 127/128 = .9921875

[MATH-06] 0x38AE F12A 7447 E99E is a 6 4-bit float written in big endian. What is the same number written in little endian?

0x9EE9 4774 2AF1 AE38

[MATH-07] What is the approximate range of an IEEE-754 double?

Nan = 0x7FF

MAX FLOAT = 0X7FEF FFFF FFFF FFFF = 2 ^2046

Min float = 2^-1022 = 0X0001 0000 0000 0000

MAX INT = 0X7FFF FFFF FFFF FFFF OR 2^63 - 1

+INFINITY = 0X7FF0 0000 0000 0000

2^-1022 to 2^2046

[MATH-08] Convert the IEEE754 number 0x0000 0000 0520 0000 to decimal?

0000 0000 0000 0000 0000 0000 0000 0000 0000 0101 0010 0000 0000 0000 0000 0000

Subnormal since 0x000 is present.

0000 0000 0000 in front shows subnormal so we know its

0000 0000 0000 0000 0000 0000 0101 0010 0000 0000 0000 0000 0000 x 2^-1022

101001.000000 x 2 ^ (-1022 + -31)

101001.0000000….. x 2 ^-1053

1 + 8 + 32 = 41 x 2 ^ -1053 = 4.28136e-316

[MATH-09] Convert the number 0x0000 0448 0000 0000 to decimal floating point.

0000 0000 0000 0000 0000 0100 0100 1000 0000 0000 0000 0000 0000 0000 0000 0000

0000 0000 0000 shows us it is subnormal. Thus

0000 0000 0100 0100 1000 x 2^-1022

10001001.0000 0000 0000 …. X 2^(-1022 + -17)

10001001.0000… x 2^-1039

1 + 8 + 128 = 137 x 2 ^-1039 = 2.325707e-311

[MATH-10] Convert the number 150 x 2^-1053 t o IEEE754 (hexadecimal)

Since 2^-1053 is less than 2 ^-1022 we know it is subnormal

Thus 1053 – 1022 = 31 know that it is -31 places

150 = 10010110

0000 0000 0000 0000 0000 0000 0000 0000 0001 0010 1100 0000 0000 0000 0000 0000

0x0000 0000 12c0 0000

[MATH-11] Convert the number 18.125 x 2^-1029 to IEEE754 (hexadecimal)

18 = 10010

.125 x 2 = 0 + .25

.25 x 2 = 0 + .50

.50 x 2 = 1 + 0

18.125 = 10010.001

2^-1029 it’s subnormal

1029 – 1023 = 7 places

0000 0000 0000 0010 0100 01 0000 0000 0000….

0x0002 4400 0000 0000

[MATH-12] What is -1738 in a 10-digit babbage machine?

00 0000 1738

0 = 9

1=8

2=7

3=6

4=5

99 9999 8261

+ 1

-------------------------------

99 9999 8262

[MATH-13] What is -2301 in a 6-digit babbage machine?

002301

997698

+ 1

997699

Convert 0.85 to IEEE754 32-bit hex. [6]

[I know that there are online websites that will do this conversion for you in a millisecond.

Showing a single answer from the web will gain you zero points. You show me sufficient

intermediate steps to convince me you know what you are doing.]

0.85

.85 x 2 = 1 + .7

.7 x 2 = 1 + .4

.4 x 2 = 0 + .8

.8 x 2 = 1 + .6

.6 x 2 = 1 + .2

.2 x 2 = 0 + .4

.4 x 2 = 0 + .8

.8 x 2 = 1 + .6

.6 x 2 = 1 + .2

.2 x 2 = 0 + .4

.4 x 2 = 0 + .8

.1101 1001 1001 1001 1001 1001 1001 x 2 ^ 0

1.1011 0011 0011 0011 0011 x 2^-1

Stored exponent: 0111 1111 = 127

127 + -1 = 126 0111 1110

126

0011 1111 0101 1001 1001 1001 1001 1001

0x3F59 9999

Convert 0x40C8 0000 into decimal floating point

0100 0000 1100 1000 0000 0000 0000 0000

1000 0001 = 1 + 2(0) + 4(0) + 8(0) + 16(0)+ 32(0) + 64(0) + 128 = 129

2^0 (1) + 2^1 (0) + 2^2(0) + 2^3(0) + 2^4(0) +

1000 0001

* 0111 1111

129 – 127 = 2 stored exponent – bias = 2

1.1001 0000 0000 …. X 2 ^2 -4 = -2

11001.0000…. x 2 ^ -2

1+ 8 + 16 = 25

25 x 2 ^ -2 = 25 x ¼ = 6.25

Convert 0.95 to IEEE-754 32-bit hex.  
.95 x 2 = 1 + .9

.9 x 2 = 1 + .8

.8 x 2 = 1 + .6

.6 x 2 = 1 + .2

.2 x 2 = 0 + .4

.4 x 2 = 0 + .8

.8 x 2 = 1 + .6

.6 x 2 = 1 + .2

.2 x 2 = 0 + .4

.4 x 2 = 0 + .8

.8 x 2 = 1 + .6

.6 x 2 = 1 + .2

.2 x 2 = 0 + .4

.4 x 2 = 0 + .8

.8 x 2 = 1 + .6

.1111 0011 0011 0011 0011 x 2^0

1.1110 0110 0110 0110 0110 x 2^-1

127 = bias = 0111 1111

True exponent

127 + -1 = 126

126 = 128(0) + 64(1) + 32(1) + 16(1) + 8(1) + 4(1) + 2(1) + 1(0)

2^6 = 64

0111 1110 = 126

0011 1111 0111 0011 0011 0011 0011 0011

0x3F73 3333

2.32 x 2 ^-130

2 ^ -126

[MATH-11] Convert the number 18.125 x 2^-1029 to IEEE754 (hexadecimal)

18 = 2^4(1) + 2^3(0) + 2^2(0) + 2^1(1) + 2^0(0) = 10010

.125 x 2 = 0 + .25

.25 x 2 = 0 + .50

.5 x 2 = 1 + .00

18.125 = 10010.001 x 2 ^ -1029

18.125 = 0010010.001 x 2^ -1029

2 ^ -1022 = 0000 0000 0000

.0010010001 x 2 ^ -1022

0x000

0000 0000 0000 0010 0100 0100 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000

0x0002 4400 0000 0000

0-9 10 = A 11= B 12= c 13 = d 14= e 15= f

MATH-13] What is -2301 in a 6-digit babbage machine?

00 2301

0 =9

1=8

2=7

3=6

4=5

997698

+ 1

--------------------------

997699

[MATH-09] Convert the number 0x0000 0448 0000 0000 to decimal floating point.

0000 0000 0000 0000 0000 0100 0100 1000 0000 0000 0000 0000 0000 0000 0000 0000

64 bits 2 ^ -1022

0000 0000 0100 0100 1000 …. X 2^-1022

X 2^ -1022 + -17

0000 0000 0100 0100 1.0000000 x 2^ -1039

10001001.0000 x 2 ^-1039

10001001 = 2^0 (1) = 1 + 8 + 128 = 137

137 x 2^-1039 = 2.325707e-311

Convert 7.125 x 2 ^-138 to a 32 bit hex

7 = 2 ^2 (1) + 2^1 (1) + 2^1 (1) = 7 111

.125 x 2 = 0 + .25

.25 x 2 = 0 + .50

.50 x 2 = 1 + .00

7.125 = 111.001

111.001 x 2^-138

32 bits subnormal since < 2 ^ -126

0 000 0000 0

111.001 x 2^-138 ; we are moving to the left 12 times

.000 0000 0011 1001 x 2 ^-126

2^ -126

0000 0000 0000 0000 0011 1001 0000 0000

0x0000 3900

Convert 0x0000 3900 to a 32 bit floating point decimal

0000 0000 0000 0000 0011 1001 0000 0000

2 ^ -126

.000 0000 0011 1001 0000 0000 x 2 ^ -126 -15

111001.0000 x 2 ^ -141

1 + 8 + 16 + 32 = 57 x 2 ^-141

[MATH-06] 0x38AE F12A 7447 E99E is a 64-bit float written in big endian. What is the same number written in little endian?

0x9EE9 4774 2AF1 AE38

0x38AE F12A 7447 E99E

If the LSB of the number is read first (processed first) then the number is

written in Little Endian

64 bit conversion

1. = 011 1111 1111 = bias

32 bit conversions

127 = 011 1111 1 = bias

Subnormal 64 bits

2 ^ -1022 = 0000 0000 0000

Subnormal 32 bits

2 ^-126 = 0000 0000 0

Xmm0 ; stored 3.1 how do you round this float towards 0

roundsd xmm1, xmm0, 3 round towards 0

roundsd xmm1, xmm0, 2 rounding towards the ceiling 3.1 -> 4

roundsd xmm1, xmm0, 1 rounds towards the floor 3.1 -> 3

roundsd xmm1, xmm0, 0 traditional -> 3.1 -> 3 3.6 -> 4

traditional method = -3.5 -4 choose even number in place tie

Swap 2 registers fast.

Xorpd xmm1, xmm2

Xorpd xmm2, xmm1

Xorpd xmm1, xmm2

Absolute value of xmm register

//Mask 7FFFFFFFFFF

Mov rax, 0x7FFFFFFFFFFFFFFF

Push rax

Movsd xmm15, [rsp]

Pop rax

andpd xmm0, xmm15

# General Knowledge [GK]

For each question in this section, clearly indicate your final answer.

[GK-01] What is the base number of a 64-bit floating-point number system? Answer in decimal.

2 ^ -1022

[GK-02] What does “immediate mode” mean? Give an example to illustrate your answer.

Immediate mode is an operand of an instruction that is a hardcoded constant

Mov r12, 100

[GK-03] What action is performed by the “ret” instruction in assembly?

Pop rip

[GK-03] Explain in technical terms exactly what c dqe does.

Converts a 32 bit number to a 64 bit number,

If number in lower 32 bits of memory is negative add 1 to the other 32 bits

If number in lower 32 bits of memory is positive add 0 to the other 32 bits

[GK-04] Who is the first programmer?

Ada lovelace

[GK-05] What action is performed by the “return” instruction in C++ and C?

Pop rbp

Pop rip

[GK-06] On the x86 architecture, which w ay does the stack grow? And the heap? What do the push/pop instructions do to the stack pointer?

Stack grows toward small addresses, Heap grow toward larger addresses.

Push makes the stack grow larger. Pop makes the stack shrink.

[GK-07] what is the Opus Magnus of all a ssembly programming? Making a rollercoaster game?

Roller Coaster Tycoon 2

[GK-08] Name all the components of a Von N eumann computer.

Bus

Processor(CPU),

Primary Storage(RAM)

Secondary Storage(SSD, disk drive, etc)

Peripherals

# GDB [GDB]

For each question in this section, show all of y our work if applicable. Clearly indicate your final answer.

[GDB-01] Suppose the following declaration creates an array in C++ language.

double myarray = new double[8];

Output the first three numbers of the array in IEEE format.

x/3xg myarray -hex

x/3fg myarray -float

[GDB-02] Suppose this array was declared: long u[6] = { 4,7,16,41,17,13}; Change the value in cell #4 to be -33.

Set var u[4] = -33

[GDB-03] What is the gdb command that w ill show the first 7 quadwords from the stack beginning with the qword at the top of the stack in IEEE format?

x/7xg $rsp

[GDB-04] Show xmm2 using default formats.

P $xmm2

[GDB-05] Show xmm0 using only IEEE754 hex f ormat for output

p/x $xmm0.v2\_int64

p/f $xmm0

[GDB-06] Show xmm5 using only decimal r epresentation of every value

p/f $xmm5

[GDB-07] Show the data in xmm12 organized as two floating point 64-bit decimal numbers.

p/f $xmm12.v2\_double

[GDB-08] Show the data in xmm13 organized as two 64-bit IEEE754 hex numbers

p/x $xmm13.v2\_double

[GDB-09] Suppose you want to look at the value in register r13 as an ascii string.

How do you output the contents of r13 as string.

x/s $r13

[GDB-10] Change the qword on top of the stack t o be -2.

Set $rsp = -2

[GDB-11] The C-string has been declared in C++ like this: char b[] = "bore";

Show how to use GDB to change the byte at c ell #1 to 'a'

Set bar b[1] = ‘a’

[GDB-12] Show the contents of memory s tarting at 0x0000 7FFF FFFF 8800 and continuing for the next 24 qwords showing the contents of each qword as an unsigned long

x/24ug 0x00007FFFFFFF8800

# Short Programs [SP]

For each question in this section, show all of y our work if applicable. Clearly indicate your final answer.

[SP-01] Make an assembly function that t akes an array and returns the minimum value. The prototype of such function is `long findMin(long arr[], l ong count);`.

15 pushes;

Mov r14, rsi //loop size

Mov r15, rdi //array

mov r13, 0 //loop counter

mov r12, [r14]

begin\_loop:

cmp r13, r14 //compare our array counter with size of array

jge finish\_loop

mov r11, [r15 + r13 \*8]

cmp r12, r11

jge skip\_part

mov r12, r11

skip\_part:

inc r13

jmp begin\_loop

finish\_loop:

mov rax, r12

15 pops

ret

[SP-02] Write an assembly program that takes an array and returns the median value. The prototype is `long findMedian(long arr[], long size);`. C an you accomplish this WITHOUT modifying the original array?

15 pushes;

Mov r14, rsi //loop size

Mov r15, rdi //array pointer

mov r13, 0 //loop counter

mov r12, 0// Median

mov rax, 0

Mov rdi, r15

Mov rsi, r14

Call sort

;Array is sorted

begin\_loop:

cmp r13, r14

jge End

Mov rax, r15

Mov r8, 2

Div r8

Cmp rdx, 0

Je even

Mov r12, [r15 + 8\*rax]

Jmp end

Even:

Mov r9, [r15 + 8\*(rax)]

Mov r10, [r15 + 8\*(rax – 1)]

Mov rax, r9

Add r10

Div 2

Mov r12, rax

End:

15 pops

[SP-03] Write an assembly program that takes an array and returns the sum o f all odd numbers.

The prototype is `long sumOdd(long arr[], l ong count);`.

15 pushes;

Mov r14, rsi //loop size

Mov r15, rdi //array pointer

mov r13, 0 //loop counter

mov r12, 0 //first odd number

begin\_loop:

cmp r13, r14

jge End

mov rax, [r15 + 8 \*r13]

mov r8, 2

div r8

cmp rdx, 0

je even

add r12, rax

Even:

Inc r13

Jmp begin\_loop

End:

Mov rax, r12

Pop 15 times

ret

[SP-04] Write a short assembly script t hat will deactivate bits 0-18 in rax and sets bit 63 to 1. The remaining bits should be a direct replica o f the ones originally in rax.

And rax, 0xFFFF FFFF FFFC 0000

Or rax, 0x8000 0000 0000 0000