Name: _____

Section	Points
I. Math Problems	/42
II. General Knowledge (SA)	/40
III. General Knowledge (MC)	/14
IV. Short Programs	/54
Total:	**/150**

Instructions:

1. This is an open-book, open-notes, open-internet exam
2. You have two hours for the exam
3. Calculators are allowed
4. Communication with classmates is NOT allowed

Notes:

- If you write the word *"blank"* in the space for the answer, you will automatically receive 25% of the credit for that question
- You may use scratch paper, however, it does not need to be submitted with the test
- Best of luck, this is a simulation so don't feel too pressured :)

I. Math Problems - Short Answer (3 points each)

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

1. Convert 34⅓ to a 64-bit IEEE-754 number. Answer in Hex.

Whole number	34 34 / 2 = 17, 0 17 / 2 = 8, 1 8 / 2 = 4, 0 4 / 2 = 2, 0 2 / 2 = 1, 0 1 / 2 = 0, 1 100010
fraction	0.33333333 0.33333333 * 2 = 0.666666666 0.66666666 * 2 = 1.333333333 0.33333333 * 2 = 0.666666666 0.66666666 * 2 = 1.333333333 0101 0101 0101 0101 0101 0101 0101 0101 0101 0101 0101 0101 0101
Binary	100010.0101 0101 0101 0101 0101 0101 0101 01010101 0101 0101 0101 0101
Scientific	1.0 0010 0101 0101 0101 0101 0101 0101 0101 01010101 0101 0101 0101 0101 x 2^5
Stored Exponent	011 1111 1111 = 1023 101 = 5 -------------------- 100 0000 0100
IEEE 754 Binary	0100 0000 0100 0001 0010 1010 1010 1010 1010 1010 1010 1010 1010 1010 1010 1010
IEEE 754 Hex	0x4041 2AAA AAAA AAAA

0x4041 2AAA AAAA AAAA

2. Convert -94.09375 to a 64-bit IEEE-754 number. Answer in Hex.

   0xC057 8C00 0000 0000

Whole number	94 94 / 2 = 47, 0 47 / 2 = 23, 1 23 / 2 = 11, 1 11 / 2 = 5, 1 5 / 2 = 2, 1 2 / 2 = 1, 0 1 / 2 = 0, 1 1011110
fraction	0.09375 0.09375 * 2 = 0.1875 0.1875 * 2 = 0.375 0.375 * 2 = 0.75 0.75 * 2 = 1.50 0.50 * 2 = 1.00 .00011
Binary	1011110.00011
Scientific	1.01111000011 x 2^6
Stored Exponent	011 1111 1111 = 1023 110 = 6 ---------- 100 0000 0101
IEEE 754 Binary	1100 0000 0101 0111 1000 0110 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
IEEE 754 Hex	0xC057 8600 0000 0000

3. Convert 0.95 to IEEE-754 32-bit hex.
   1 + 8 + 23 = 32

Whole number	0
fraction	.95 0.95 * 2 = 1.90 0.90 * 2 = 1.80 0.80 * 2 = 1.60 0.60 * 2 = 1.20 0.20 * 2 = 0.40 0.40 * 2 = 0.80 0.80 * 2 = 1.60 0.60 * 2 = 1.20 . 1111 0011 0011 0011 0011 001
Binary	. 1111 0011 0011 0011 0011 001
Scientific	1.111 0011 0011 0011 0011 001 * 2^-1
Stored Exponent	0111 1111 = 127 -0000 0001 = -1 --------------- 0111 1110 = 126
IEEE 754 Binary	0011 1111 0111 0011 0011 0011 0011 0011
IEEE 754 Hex	0x3F73 3333

4. Convert 0x3FB9 0000 0000 0000 to a decimal floating-point number. The answer should be in fraction notation.

Binary	0011 1111 1011 1001 0000 .. 0000
Binary	0011 1111 1011 1001 0000 .. 0000
Exponent	011 1111 1011 = 1019 -011 1111 1111 = 1023 ---------- -000 0000 0100 = -4 (true exponent)
Significand Mantissa	 1001
Scientific	1.1001 x 2^-4
Fractions Method 1	(1 + ½ + 0 + 0 + 1/16) * 1/16 1/16 + 1/32 + 1/256 16/256 + 8/256 + 1/256 25/256
Fractions Method 2	11001 x 2^-8 (move decimal by 4 spots to the right) (16 + 8 + 0 + 0 + 1)/256 25/256

100001 = 1.00001 x 2^5

1.00001 x 2^5

Scientific	1.10001 x 2^4
Fractions Method 1	(1 + ½ + 0 + 0 + 0 + 1/32) * 16 (1 + 1/2 + 1/256 16/256 + 8/256 + 1/256 25/256
Fractions Method 2	110001 x 2^-1 (move decimal by 5 spots to the right) (16 + 8 + 0 + 0 + 0 + 1)/256 25/256

5. What is positive infinity in 64-bit IEEE-754 format? Answer in Hex.

Hex	0x7FF0 0000 0000 0000
Binary	0111 1111 1111 0000 0000 0000 … 0000

6. What is negative infinity in 64-bit IEEE-754 format? Answer in Hex.

Hex	0xFFF0 0000 0000 0000
Binary	1111 1111 1111 0000 0000 0000 … 0000

7. Convert -985 to hex suitable for 32-bit registers.

Convert to Binary	985 / 2 = 492, 1 492 / 2 = 246, 0 246 / 2 = 123, 0 123 / 2 = 61, 1 61 / 2 = 30, 1 30 / 2 = 15, 0 15 / 2 = 7, 1 7 / 2 = 3, 1 3 / 2 = 1, 1 1 / 2 = 0, 1
Binary	0000 0000 0000 0000 0000 0011 1101 1001
1's complement	1111 1111 1111 1111 1111 1100 0010 0110
2's complement	1111 1111 1111 1111 1111 1100 0010 0111
Hex	0xFFFF FC27

8. Convert -3212 to 64-bit integer hex form. Show the final answer in little-endian (for people who naturally read from right to left)

Convert to Binary	3212 / 2 = 1606, 0 1606 / 2 = 803, 0 803 / 2 = 401, 1 401 / 2 = 200, 1 200 / 2 = 100, 0 100 / 2 = 50, 0 50 / 2 = 25, 0 25 / 2 = 12, 1 12 / 2 = 6, 0 6 / 2 = 3, 0 3 / 2 = 1, 1 1 / 2 = 0, 1	
Binary	0000 0000 0000 0000 0000 1100 1000 1100	0x0000 0000 0000 0C8C
1's complement	1111 1111 1111 1111 1111 0011 0111 0011	0xFFFF FFFF FFFF F373
2's complement	1111 1111 1111 1111 1111 0011 0111 0100	0xFFFF FFFF FFFF F374
Hex (big endian)	0xFFFF FFFF FFFF F374	0xFFFF FFFF FFFF F374
Hex (little endian	0x74F3 FFFF FFFF FFFF	0x74F3 FFFF

9. Suppose you are attempting to convert 3932 to binary. What would the third step look like?

3932 / 2 = 1966, remainder = 0
1966 / 2 = 983, remainder = 0
983 / 2 = 491, remainder = 1

10. Calculate 0x00402A43 - 0x0023E07F in hex.

Not on the on the test

Subtraction	Addition with 2's Complement
0x00402A43 -0x0023E07F	0x00402A43 -0x0023E07F
3F 93 0x00402A43	0x00402A43 +0xFFDC1F81

-0x0023E07F ----------- 1C49C4	-----------
	11 1 0x00402A43 +0xFFDC1F81 ----------- 1001C49C4 001C49C4

11. Represent 0xFFFF FFFF 075C 43DE in little endian.

Big endian	0xFFFF FFFF 075C 43DE
Little Endian	0xDE43 5C07 FFFF FFFF

12. What is the smallest signed integer in 32-bit two's complement? Answer in Hex.

Binary	1000 0000 0000 0000 0000 0000 0000 0000
Hex	0x8000 0000
Decimal	-2^31

Binary	0111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111
Hex	0x7FFF FFFF
Decimal	2^31 -1

13. What is the smallest twos-complement 32-bit integer? Give your answer in decimal.

    SI leader will check on this one

Binary	1000 0000 0000 0000 0000 0000 0000 0000
Hex	0x8000 0000
Decimal	-2^31

14. What is the hex representation of the largest signed integer in a 48-bit register?

Binary	0111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 47 ones (48 - 1)
Hex	0x7FFF FFFF FFFF
Decimal	2^47 - 1

II. General Knowledge - Short Answer (2 points each)

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

1. According to the professor, what is the Opus Magnus of all assembly programming?

   Roller Coaster Tycoon 2

2. According to the professor, who is the King of Assembly?

   Chris Sawyer

   King of Hardware: Charles Babbage

3. What is the full URL leading to the complete legal document written in ASCII text used in the software license promoted in this course for application software?
   https://www.gnu.org/licenses/gpl-3.0.txt

4. What is the official name of the Linux shell available for Windows 10 machines?

Windows Subsystem for linux

5. What is the backend boundary and what is the frontend boundary of the activation record? Specify each part of the answer with “back” or “front”.

   Back is rbp, front is rsp (current record)
   Back is rbp, front is rip (previous records)

6. Explain precisely what cdqe does.

   Takes the value of eax (lower 32 bits of rax) and widens the value to 64 bits. This will preserve the sign of the 32 bit value. -1 (32 bit = 0xFFFFFFFF) will be come -1 (64 bit - 0xFFFFFFFFFFFFFFFF). It does this by taking the 31st bit of eax and copying it into each of the bits 32 to 63 of rax.

7. A C++ (or C) function has declared float weights[40]; Output the contents of the first 12 dwords of the array in decimal.

   x/12fw weights, p/f *weights@12 (c only)

8. Show the contents of memory starting at 0x0000 7FFF FFFF 8800 and continuing for the next 18 qwords showing the contents of each qword as an unsigned long.

   x/18ug 0x00007FFFFFFF8800

9. Change the dword at memory address 0x0000 0FFF 8000 4400 to be 330

   set {int} 0x00000FFF80004400 = 330

   Don’t know how to specify the size

10. Change the qword on top of the stack to be -2.

    set *(long)$rsp = -2
    set {long} $rsp = -2

11. What is the rule that determines if a written number or a stored number is in Little Endian format?

    Not sure what this means. Unless it is the natural reading order.
    If number is in text, and the natural order is left to right, then number is in big endian. But if natural order is right to left, then the number is in little endian.

12. What are the names (or acronyms) of the "components" of a modern X86 microprocessor?

    GPR, FPU, SSE, AVX
    ALU, Cache

13. Name all the components of a Von Neumann computer.

    Processor, primary storage (RAM), secondary storage (HDD/SSD), peripherals (keyboard, monitor), Bus that connects them all.

14. In the normal order of things, what do we call the first byte of a big-endian number?

    Most significant byte

15. What is the defining property of a "little-endian" when speaking of numbers?

    The least significant byte is listed stored at the lowest memory address.

16. In the C++ compilation command, there is a switch -c. What is the point of that switch?

    This compiles the source file only into object code (.o). It does not link or create an executable.

17. What happens if you include the S parameter in the compilation command of a C++ function.

    Generates assembly output that does not look like NASM assembly

18. Most modern computers have a single stack at one end of memory. As we move away from the stack to the other end of memory what is the first important block of data we encounter?

    Heap

19. What is the name of the integer number system used in most modern computers?

    2's complement

20. On the x86 architecture, which way does the stack grow? And the heap? What do the push/pop instructions do to the stack pointer?

    Stack grows down (lower addresses as items are pushed on the stack), the heap grows up (larger addresses as more memory is allocated.). Push will decrease the value of the stack pointer because the stack grows down, pop will increase the value of the stack pointer.

21. What is the relationship between assembly language and machine language?

    Assembly language is a higher level language than machine and make it easier to write a program that targets the specific architecture of the machine.

    Assembly language is human readable machine language.

22. 1. Suppose char b[ ] = "Hello"; has been declared.
    Show the GDB command that will output b in ascii numbers.

    x/5cb b

23. Refer to the array of the preceding question. What is the GDB command that will display the starting address of the array?

    p/x &b

24. 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

25. The C-string has been declared in C++ like this: char b[] = "bore";
    Show how to use GDB to change the byte at cell #1 to 'a'.

    Set var b[1] = `a`

26. Show how to output the c-string in the previous question using gdb.
    p/s b

27. Starting at this address 0x7fffffffde80 show how to output the 15 dwords in hex. (Caution: watch for 'dword', which is 32 bits.)
    x/15xw 0x7fffffffde80

III. General Knowledge - Multiple Choice (2 points each)

For each question in this section, circle **1** answer. Choose the best answer.

1. Which of these has the highest address?
   - **a. <u>.bss segment</u>**
   - b. .data segment
   - c. Operating system segment
   - d. Cache memory

2. Which of these has a lower address than any of the others?
   - a. .bss segment
   - b. .data segment
   - ***c. Executing Code***
   - d. Cache Memory

3. Which register would be used to pass the second functional argument in 64-bit Linux?

   a. ***rsi***
   b. rdi
   c. rax
   d. r12

4. A DWORD represents how many bytes?
   a. 2 bytes
   b. 4 bytes
   c. 8 bytes
   d. 16 bytes

5. Which jump instruction is equivalent to je?
   a. jz
   b. jle
   c. jne
   d. js

6. Which register points to the current top of the stack?
   a. rsp
   b. rbp
   c. rsi
   d. rdx

7. What is the name of the section where the code is placed?
   a. .data
   b. .text
   c. .bss

IV. Short Programs - Short Answer (6 points each)

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

1. Your program contains a `printf` block like this:

   ```
   mov rax, 0
   mov rdi, computation
   mov rsi, r8
   call printf
   ```

   While testing you discover a problem, `printf` changes your values stored in `rdx`, `r8`, and `r9`. At the moment there are no unused registers where `rdx`, `r8`, and `r9` can be backed up. Provide an example of how you can fix this issue.

   push rdx

```
push r8
push r9
push qword -1  ; for preserving 16 byte alignment on stack
mov rax, 0
mov rdi, computation
mov rsi, r8
call printf
pop rax
pop r9
pop r8
pop rdx
```

2. Suppose there are signed integers stored in `r8` and `r9`. Complete the assembly code below such that the code fragment divides the numerator r8 by the denominator `r9` and output the statement: `"The quotient is 23"`

   We don't need this

3. Consider the following code fragment. What are the contents of the `rax`, `rbx`, and `r14` registers after the code fragment has been executed? Show your work.

```
mov r15, 21          r15 = 21
mov r14, 18          r14 = 18

mov rax, r15         rax = 21
add r14, 2           r14 = 20
add r15, 6           r15 = 27
mov rbx, r14         rbx = 20
add rbx, 4           rbx = 24
mov rax, r15         rax = 27
```

4. Consider the following code fragment. What are the contents of the `rax` and `rbx` after the code fragment has been executed? Show your work.

```
mov rax, 4
mov rbx, 16
mov rbx, [rax]
mov [rbx], 56
```

----------Memory----------

ADDRESS	VALUE
000000	6543210
000004	**5189784**
000008	1698791
00000C	9816517
000010	9816875
000014	5498156

Rax = 4, rbx = **5189784**

[rax] = **5189784**
[rbx] = 56

5. We want a function that copies the data of one array to a second array. Assume `a` and `b` are two arrays of long ints. Both arrays have the same size. Assume `s` is the number of valid data in `a`.
   a. What is the C++ prototype of a function that copies the first s numbers from `a` to `b`?

      extern "C" void copy(long a[], long b[], long s)

   b. The following facts are given:
      books is an array of 200 quadwords declared in .bss
      documents is an array of 200 quadwords declared in .bss
      `r12` contains an integer, 0 <= `r12` < 200

`r12` is the number of valid data in books

Show the block of asm instructions for calling the function that will copy books to documents.

```
; r12 is the number of books
mov r13, 0

beginloop:
    cmp r13, r12
    je endloop
    mov rax, [books + 8 * r13]
    mov [documents + 8 * r13], rax
    inc r13
    jmp beginloop

endloop:
```

6. If the call to a function is always segfaulting what should the programmer try first to fix this issue?

   Add a push qword 0 before the function call and a pop rax after the function call.

7. Assume that oranges and grapes are arrays of size 100, also assume that they are already declared in the .bss section. You want to create an assembly code that will copy the first 30 values of oranges into the first 30 values of grapes. Translate the C++ code below to assembly.

```
for(int i = 0; i < 30; i++) {
      grapes[k] = oranges[k];
}
```

```
mov r13, 0
mov r12, 30
beginloop2:
    cmp r13, r12
    je endloop2
    mov rax, [oranges + 8 * r13]
    mov [grapes + 8 * r13], rax
    inc r13
    jmp beginloop2

endloop2:
```

```
0x0000000000000000,
0x0010000000000000,
0xFFF0000000000000 = -inf,

0x7FF0000000000000 = inf,
0x7FF0000000000001 = nan,
...
0x7FFFFFFFFFFFFFFF = nan
```