CS210 Fall 2023: PS3A

Instructions

For all multiple choice questions, fill **ONE AND ONLY ONE circle**. Be sure to fill the circle in completely.
For all the questions, we encourage you to log in into the provided UNIX environment and explore your answers. For some questions, you must use the UNIX environment to answer them.

If you use checkmarks or other symbols, the auto-grader may not be able to process your answer and will assign you a grade of zero.

All pages must have your name and id written on it. Unidentified pages will not be graded.

First Name: Jae Hong Last Name: Lee

BU ID: U21565203

First Name: Jae Hong Last Name: Lee BU ID: U27565203

Multiple Choice

1. (1 point) 'je' is considered an ALU operation.
   - ◯ True.
   - ● False.

2. (1 point) What is the purpose of the Instruction Register (IR)?
   - ● The Instruction Register stores the opcode to be executed.
   - ◯ The Instruction Register stores where in memory the opcode is located.
   - ◯ The Instruction Register stores the line number of the line of code in a program currently being executed.
   - ◯ The Instruction Register stores the previously executed opcode.

3. (1 point) Little Endian ordering...
   - ◯ orders bytes from least significant to most significant.
   - ◯ applies to values in memory.
   - ◯ is a type of ordering that applies to multi-byte values created from single byte values.
   - ● All of the above.
   - ◯ None of the above.

4. (1 point) What stage occurs after the 'execute' phase of the CPU loop completes?
   - ◯ None - the loop breaks while the CPU awaits execution of another program.
   - ◯ The Decode stage.
   - ● The Fetch stage.
   - ◯ The Reset stage.

First Name: Jae Hong Last Name: Lee BU ID: U27565203

5. (1 point) The linker...
   - ◯ is a tool that can create binary object files.
   - ◯ takes in assembly source files.
   - ◯ requires at least two files as an input.
   - ◯ needs no knowledge of the current operating system.
   - ◯ All of the above.
   - ● None of the above.

6. (1 point) On Intel CPUs, the EFLAGS register...
   - ◯ is updated with conditional jump operations.
   - ● is updated with ALU operations.
   - ◯ is a general purpose register, or GPR.
   - ◯ All of the above.
   - ◯ None of the above.

7. (1 point) Aside from the assembly source code, what other information is provided to the assembler?
   - ● Instruction Set Architecture
   - ◯ Memory Mapping Configuration
   - ◯ The operating system's syscall definitions
   - ◯ The system's file streams
   - ◯ All of the above
   - ◯ None of the above

First Name: Jae Hong Last Name: Lee BU ID: U27565203

Basic Byte Representation

0000 0
0001 1
0010 2
0011 3
0100 4
0101 5
0110 6
0111 7
1000 8
1001 9
1010 A
1011 B
1100 C
1101 D
1110 E
1111 F

The following is the gdb dump of 64 bytes of memory in base 2 notation. Using this data please fill in the following tables.

0x402000:	01001000	01101111	01100010	01100010	01111001	01110100	01100101	00100001
	4 8	6 F	6 2	6 2	7 9	7 4	6 5	2 1

8. (4 points) Write the values as single byte values in **hex** notation.

0x402000	0x48	0x6F	0x62	0x62	0x79	0x74	0x65	0x21

9. (4 points) As 2-byte little endian values in **hex** notation.

0x402000	0x6F48	0x6262	0x7479	0x2165

10. (4 points) As 4-byte little endian values in **hex** notation.

0x402000	0x62626F48	0x21657479

11. (4 points) As an 8-byte little endian value in **hex** notation.

0x402000	0x2165747962626F48

12. (4 points) Finally using the provided ASCII Table please fill in the table below translating each byte into an ascii character.

0x402000	H	o	b	b	y	t	e	!

First Name: Jue Hong Last Name: Lee BU ID: U27565203

Assembly Fragments

Given the code and list of gdb commands below, answer the following questions. Assume the code has been assembled and linked correctly to produce a binary, after which gdb is used with the binary to run the given gdb commands.

Remember to use Little Endian byte ordering for multi-byte values when displayed as single bytes

13. Assembly code for `muldiv.S`:

```
1         .intel_syntax noprefix
2         .section .data
3
4 mulvalue:
5         .quad 0x3
6 divvalue:
7         .quad 0x2
8
9         .section .text
10        .global _start
11
12 _start:
13        mul QWORD PTR [mulvalue]
14        dec rax
15        dec rax
16        div QWORD PTR [divvalue]
17        mov QWORD PTR [mulvalue], rax
18        cmp rax, 9
19        jl B
20 A:
21        jmp C
22 B:
23        add rax, 6
24 C:
25        int3
```

First Name: Jae Hong Last Name: Lee BU ID: U27565203

Gdb commands used with the binary `muldiv` produced from `muldiv.S`.

```
1  file muldiv
2  set disassembly-flavor intel
3  b _start
4  run
5  delete 1
6  set $rax = 6
7  si
8  si
9  si
10 si
11 si
12 si
13 si
14 si
15 p /x $rax
16 p /x $pc
17 x /1xg &mulvalue
18 x /8xb &mulvalue
19 quit
```

First Name: Joe Hoz Last Name: Lee BU ID: U27565203

Addtionally this is the gdb output for the gdb command at line 3 of the above commands:

```
1  (gdb) x/10i _start
2  0x401000 <_start>:    mul    QWORD PTR ds:0x402000
3  0x401008 <_start+8>:  dec    rax
4  0x40100b <_start+11>:        dec    rax
5  0x40100e <_start+14>:        div    QWORD PTR ds:0x402008
6  0x401016 <_start+22>:        mov    QWORD PTR ds:0x402000,rax
7  0x40101e <_start+30>:        cmp    rax,0x9
8  0x401022 <_start+34>:        jl     0x401026 <B>
9  0x401024 <A>:          jmp    0x40102a <C>
10 0x401026 <B>:          add    rax,0x6
11 0x40102a <C>:          int3
```

(a) (2 points) Value displayed for rax on line 15 of gdb commands: 0xe

(b) (2 points) Value displayed for pc on line 16 of gdb commands: 0x40102a

(c) (2 points) Value displayed for line 17 of gdb commands: 0x0000000000000008
(`x/1xg &mulvalue` means display one 64bit value at the address in memory of the `mulvalue` symbol in hex notation)

(d) (1 point) Values displayed on line 18 of gdb commands:

08

00

00

00

00

00

00

00