Midterm II

Graded

Student

Jae Hong Lee

Total Points

58 / 62 pts

Question 1

Q1 — 1 / 1 pt

✔ + 1 pt Correct

Question 2

Q2 — 1 / 1 pt

✔ + 1 pt Correct

Question 3

Q3 — 1 / 1 pt

✔ + 1 pt Correct

Question 4

Q4 — 0 / 1 pt

✔ + 0 pts Incorrect: correct answer is true

Question 5

Q5 — 0 / 1 pt

✔ + 0 pts Incorrect: correct answer is none of the above

Question 6

Q6 — 1 / 1 pt

✔ + 1 pt Correct

Question 7

Q7 — 0 / 1 pt

✔ + 0 pts Incorrect: correct answer is first choice

Question 8

Q8 **1** / 1 pt

✔ **+ 1 pt** Correct

Question 9

Q9 **1** / 1 pt

✔ **+ 1 pt** Correct

Question 10

Q10 **0** / 1 pt

✔ **+ 0 pts** Incorrect: correct answer is third choice

Question 11

Q11 **1** / 1 pt

✔ **+ 1 pt** Correct

Question 12

Q12 **4** / 4 pts

✔ **+ 4 pts** Correct

Question 13

Q13 **4** / 4 pts

✔ **+ 4 pts** Correct, with previous errors. Correct answer is `0x2d3b, 0x5729, 0x4c61, 0x4f44`

Question 14

Q14 **4** / 4 pts

✔ **+ 4 pts** Correct

Question 15

Q15 **4** / 4 pts

✔ **+ 4 pts** Correct

Question 16

Q16 **4** / 4 pts

✔ **+ 4 pts** Correct

Question 17

Q17 **7** / 7 pts

17.1 a **2** / 2 pts

✔ **+ 2 pts** Correct

17.2 b **2** / 2 pts

✔ **+ 2 pts** Correct

17.3 c **2** / 2 pts

✔ **+ 2 pts** Correct

17.4 d **1** / 1 pt

✔ **+ 1 pt** Correct

Question 18

Q18 **7** / 7 pts

18.1 a **3** / 3 pts

✔ **+ 3 pts** Correct

18.2 b **2** / 2 pts

✔ **+ 2 pts** Correct

18.3 c **2** / 2 pts

✔ **+ 2 pts** Correct

Question 19

Q19 **7** / 7 pts

19.1 a **3** / 3 pts

✔ **+ 3 pts** Correct

19.2 b **2** / 2 pts

✔ **+ 2 pts** Correct

19.3 c **2** / 2 pts

✔ **+ 2 pts** Correct

Question 20

Q20 **6** / 6 pts

20.1 **a** **4** / 4 pts

✔ **+ 4 pts** Correct

20.2 **b** **2** / 2 pts

✔ **+ 2 pts** Correct

Question 21

Q21 **4** / 4 pts

21.1 **a** **1** / 1 pt

✔ **+ 1 pt** Correct

21.2 **b** **1** / 1 pt

✔ **+ 1 pt** Correct

21.3 **c** **1** / 1 pt

✔ **+ 1 pt** Correct

21.4 **d** **1** / 1 pt

✔ **+ 1 pt** Correct

CS210 Computer Systems, Fall-2023
Midterm-II B

Instructions

This is a closed book and closed notes exam. NO ELECTRONIC DEVICES. For all multiple choice questions fill in ONE and ONLY ONE circle. Fill the circle in completely.

If you use check marks or other symbols the auto-grader may not be able to process your answer and will assign you a grade of zero and there will be no regrading.

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

This exam has 21 questions, for a total of 62 points.

First Name: Jae Hong Last Name: Lee

BU ID: U29565203

First Name: Jue Hong Last Name: LEE BU ID: U29565203

PART A: True/False and Multiple Choice

1. (1 point) Which register contains the length of data to write in the write syscall?
   - ○ rax
   - ○ rdi
   - ○ rsi
   - ● rdx

   pc

2. (1 point) The program counter is:
   - ○ A special register that counts the number of mov instructions. X
   - ○ Typically used to track the bottom address of the stack. X
   - ○ A register used to compute an effective address. X
   - ○ A special register that is composed of several single-bit condition flags. X
   - ○ All of the above X
   - ● None of the above

3. (1 point) The stack area of memory

   push

   - ○ is only used when the program makes a system call X
   - ○ is implemented as a First-In-First-Out (FIFO) data structure X
   - ● can be used to store the local variables of a function call
   - ○ All of the above X
   - ○ None of the above

4. (1 point) The assembler is a tool for generating binary encoded opcodes.
   - ○ True

   objective file.

   - ● False

First Name: Jue Hong Last Name: Lee BLID: U29565203

5. (1 point) The INTEL stack pointer register (RSP):
   - ○ stores the value on the top of the stack
   - ○ stores the address of the next instruction to execute
   - ● stores the address of the value at the bottom of the stack
   - ○ all of the above
   - ○ none of the above

6. (1 point) Endianness
   - ○ instructs the CPU to always print bytes from least significant to most significant X
   - ○ allows us to use ".intel-syntax noprefix" X
   - ● is a convention for referring to bytes of a multi-byte value O
   - ○ all of the above X
   - ○ none of the above

7. (1 point) The linker produces binary
   - ○ an executable object file that can be loaded and run X
   - ● gdb directives
   - ○ a set of relocatable object files X
   - ○ all of the above X
   - ○ none of the above

8. (1 point) The Virtual Address Space
   - ○ is the same size as the Physical Address Space X
   - ○ is shared by all running programs X
   - ● allows a program to have a private and independent view of memory
   - ○ all of the above X
   - ○ none of the above

9. (1 point) Which of the following is not a typical Intel GNU Assembly Instruction format:
   - ○ mnemonic <destination> 9
   - ○ mnemonic
   - ○ mnemonic <destination>, <source>
   - ● mnemonic <destination>, <destination>
   - ○ mnemonic <destination>, <source A>, <source B>

First Name: Jae Hong Last Name: Lee BUID: U27565203

10. (1 point) The following code:

```
1         .intel_syntax noprefix
2         .section .text
3         .global _start
4 _start:                          rdi : ∅ 2
5         xor rdi, rdi
6         jmp F
7 R1:
8         jmp F
9 R2:
10        mov rax, 60     # LINUX: exit system call 60
11        syscall
12
13 F:          ∅
14        add rdi,2
15        jz ∅ R1
16        jmp R2
```

- (●) will run forever
- ( ) will not raise a runtime error ∅
- ( ) will exit with rdi equal to 2 0
- ( ) will exit with edi (-f100) equal to 4 X
- ( ) All of the above X
- ( ) None of the above

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

4. (1 point) The following code:

```
1          .intel_syntax noprefix
2
3          .section .data
4  x:
5          .quad 2
6
7  y:
8          .quad 4
9
10         .section .text
11         .global _start
12 _start:
13         mov rax, 1
14         mov rbx, QWORD PTR [y]
15         mov QWORD PTR[y], QWORD PTR [x]
16         mov QWORD PTR [x], rbx
17
18         /* Call OS EXIT system call */
19         mov rax, 60  # move exit system call number 60 into rax
20         mov rdi, 0   # move exit status code into rdi
21         syscall      # call OS kernel (process will terminate)
```

x: 74
y: 42

rax: 1
rbx: 4

data
two variable cause error.

- (●) will cause the assembler to throw an error
- ( ) will exit with x equal to 4 and y equal to 4 X
- ( ) will exit with x equal to 4 and y equal to 2
- ( ) will exit with rax equal to 60 X
- ( ) All of the above X
- ( ) None of the above

First Name: Jue Hong Last Name: Lee BUID: U27565203

PART B

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.

1 0x402000:	00111011	00101101	00101001	01010111	01100001	01001100	01000100	01001111
	3 b	2 d	2 9	5 7	6 1	4 C	4 4	4 F

12. (4 points) Write the values as single byte values in **hex** notation.

0x402000	3b	2d	29	57	61	4c	44	4f

13. (4 points) As 2-byte little endian values in **hex** notation.

0x402000	2d3b	5729	4c61	4f44

14. (4 points) As 4-byte little endian values in **hex** notation.

0x402000	5729 2d3b	4f44 4c61

15. (4 points) As an 8-byte little endian value in **hex** notation.

0x402000	4f44 4c61 5729 2d3b

16. (4 points) Finally using the provided ASCII Table please fill in the table below translating each byte into an ascii character.

0x402000	;	-	)	W	a	L	D	O

First Name: Jue Hong Last Name: Lee BUID: U27565203

PART C: 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

17. An assembly fragment using the `inc`, `dec` and `mov` instructions. Assembly code for `frag2.s`:

```
1         .intel_syntax noprefix
2
3         .section .data
4 value:
5         .quad 0xce
6
7         .section .text
8         .global _start
9
10 _start:
11        mov rbx, QWORD PTR [value]
12        inc rdx
13        dec rbx
14        mov QWORD PTR [value], rbx
15        int3
```

value : 0xce 0xcd
rbx : 0xce 0xcd
rdx : 1

Gdb commands used with the binary `frag2` produced from `frag2.s`.

```
1  file frag2
2  set disassembly-flavor intel
3  x/5i _start
4  b _start
5  run
6  delete 1
7  si
8  si
9  si
10 si
11 p /x $rbx
12 p /x $pc
13 x/1xg &value
14 x/8xb &value
15 quit
```

0xcd
int 3
0xcd

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

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

```
1  (gdb) x/5i _start
2     0x401000 <_start>:     mov     rbx,QWORD PTR ds:0x402000
3     0x401008 <_start+8>:   inc     rdx
4     0x40100b <_start+11>:  dec     rbx
5     0x40100e <_start+14>:  mov     QWORD PTR ds:0x402000,rbx
6     0x401016 <_start+22>:  int3
```

(a) (2 points) Value displayed for rbx on line 11 of gdb commands: 0xcd

(b) (2 points) Value displayed for pc on line 12 of gdb commands: 0x401016

(c) (2 points) Value displayed for line 13 of gdb commands 0xcd
(`x/1xg &value` means display one 64bit value at the address in memory of the `value` symbol in hex notation):

(d) (1 point) Values displayed on line 14 of gdb commands:

0xcd

0x00

0x00

0x00

0x00

0x00

0x00

0x00

First Name: Jue Hong Last Name: Lee BUID: U27565203

18. A conditional based assembly fragment

Assembly code for `frag4.s`.

```
1   .intel_syntax noprefix
2
3   .section .data
4   X:
5   .quad 3
6
7   .section .text
8   .global _start
9   _start:
10  mov rcx, 0x3
11  mov rbx, QWORD PTR [X]
12  cmp rax, rbx
13  jge A
14  dec rcx
15  jmp B
16  A:
17  mov rcx, 0x5
18  inc rcx
19  B:
20  add rax, 3
21  add rcx, 5
22  int3
```

rax: ~~3~~ 6
X: 3
rcx: ~~0x3~~ ~~0x5~~ ~~0x6~~ 11B 10A
rbx: 3
jge A greater or equal

Gdb commands used with the binary `frag4` produced from `frag4.s`.

```
1  file frag4
2  b _start
3  run
4  set $rax = 3
5  delete 1
6  c
7  p /x $rax
8  p /x $rbx
9  p /x $rcx
10 quit
```

6
3
11

Note: the "c" gdb command on line 6 continues execution until the binary stops at the "int3" instruction on line 22 of the source code.

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

Please fill in the following:

(a) (3 points) Value displayed for rax on line 7 of gdb commands: 0x6

(b) (2 points) Value displayed for rbx on line 8 of gdb commands: 0x3

(c) (2 points) Value displayed for rcx on line 9 of gdb commands: 0x6

First Name: Jue Hong Last Name: Lee BUID: 027565203

19. A loop based assembly fragment.

Assembly code for `frag6.s`:

```
1         .intel_syntax noprefix
2
3         .section .data
4  data:                0   1   2   3   4
5         .quad 0x1001, 0xA132, 0x5CF1, 0x1966, 0x7906
6         .quad 0xDF42, 0x7152, 0xCDA9, 0xEE54, 0x1212
7                       5     6     7     8     9
8         .section .text
9         .global _start                    rbx: data
10                                          rcx: 4 5 6 7 8
11 _start:                                  rax: 0x7906 DF42 7152 CDA9
12        mov rbx, OFFSET [data]
13        mov rcx, 4        402000
14 A:                       rcx 4
15        cmp rcx, 8
16        je  done
17        mov rax, QWORD PTR [rbx + 8*rcx]
18        inc rcx
19        jmp  A
20 done:
21        int3
```

Gdb commands used with the binary `frag6` produced from `frag6.s`.

```
1 file frag6
2 b _start
3 run
4 delete 1
5 c
6 p /x $rax   0x CDA9
7 p /x $rbx   402000
8 p /x $rcx   8
9 quit
```

Assume the address of symbol `data` is 0x402000.

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

Please fill in the following:

(a) (3 points) Value displayed for rax on line 6 of gdb commands: 0xCDA9

(b) (2 points) Value displayed for rbx on line 7 of gdb commands: 0x402000

(c) (2 points) Value displayed for rcx on line 8 of gdb commands: 0x8

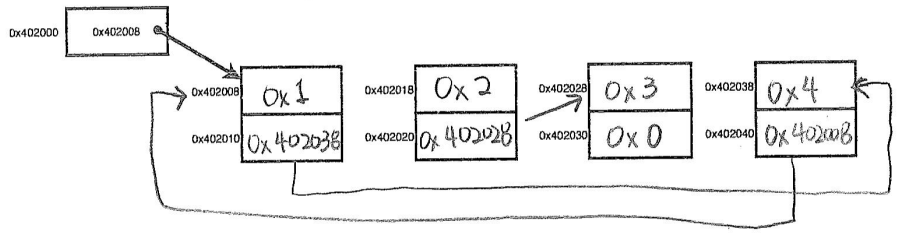
First Name: Jue Hong Last Name: Lee BU ID: U21565203

20. Given the following memory contents gathered with gdb, please answer the following questions:

```
1  (gdb) x/8i _start
2     0x401000 <_start>:    xor     rdi,rdi
3     0x401003 <_start+3>:  mov     rbx,QWORD PTR ds:0x402000
4     0x40100b <A>:         add     rdi,QWORD PTR [rbx]
5     0x40100e <A+3>:       mov     rbx,QWORD PTR [rbx+0x8]
6     0x401012 <A+7>:       cmp     rbx,0x0
7     0x401016 <A+11>:      jne     0x40100b <A>
8     0x401018 <A+13>:      mov     rax,0x3c
9     0x40101f <A+20>:      syscall
10 (gdb) x/1gx 0x402000
11    0x402000:    0x0000000000402008
12 (gdb) x/8gx 0x402008
13 0x402008:       0x0000000000000001      0x0000000000402038
14 0x402018:       0x0000000000000002      0x0000000000402028
15 0x402028:       0x0000000000000003      0x0000000000000000
16 0x402038:       0x0000000000000004      0x0000000000402008
```

rdi: 0 1 5 6
rbx: 402008 → 402010 → 402038 → 402028 → 402008 → 402010 → 402038

(a) (4 points) Please update the following diagram. Be sure to fill all boxes **and include arrows to indicate address relationships.**



(b) (2 points) What will be the value of rdi when the program exits (the syscall on line 9 is executed)? If the program does not exit then your answer should be "NONE".

rdi= NONE

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

21. Given the following incomplete LOWER function from the PS4B calculator assignment.

```
1         .intel syntax noprefix
2         .section .text
3         .global LOWER
4   LOWER:
5         # INPUTS: rax -> x
6         #         rbx -> contains the address of a quad word
7         #                that is the address of the string
8         # OUTPUTS: x = x + length of string
9         #           When done any lower case letters should be
10        #           translated to an upper case letter
11        mov   rdx, QWORD PTR [rbx]
12        test  rdx, rdx
13        jz    L3 ----------------
14        xor   rdi, rdi
15  L1:
16        mov, r8b, BYTE PTR [rdx+rdi] ----------
17        test  r8b, r8b
18        jz    L3
19        cmp   r8b, 'A'
20        jb    L2
21        cmp   r8b, 'Z'
22        ja    L2
23        add   r8b, 'a'-'A'
24        mov   BYTE PTR [rdx + rdi], r8b
25  L2:
26        ------------------------
27        jmp   L1
28  L3:
29        ------------------------
30        add   rbx, 8      # update rbx and we have finished this
31                          # command
32        ret
```

rdx : string upp -> lower.
rdi : 0
done
if below.
if above
20
61-41

Provide the missing code for the blanks, below:

(a) (1 point) Line 13: L3

(b) (1 point) Line 16: mov r8b, BYTE PTR [rdx+rdi]

(c) (1 point) Line 26: inc rdi

(d) (1 point) Line 29: add rax, rdi