CS 210 Midterm 1

Jeong Yong Yang

TOTAL POINTS

32 / 40

QUESTION 1

1 VNA: Overview: a **0 / 1**

✓ **- 1 pts** Incorrect. Correct answer is "A set of locations ...".

QUESTION 2

2 VNA: Overview: b **1 / 1**

✓ **- 0 pts** Correct

QUESTION 3

3 VNA: Overview: c **1 / 1**

✓ **- 0 pts** Correct

QUESTION 4

4 VNA: Overview: d **1 / 1**

✓ **- 0 pts** Correct

QUESTION 5

5 VNA: Overview: e **1 / 1**

✓ **- 0 pts** Correct

QUESTION 6

6 VNA: ALU and Data Rep: a **0 / 1**

✓ **- 1 pts** Incorrect. Correct answer is "a sequence of bytes that..."

QUESTION 7

7 VNA: ALU and Data Rep: b **1 / 1**

✓ **- 0 pts** Correct

QUESTION 8

8 VNA: ALU and Data Rep: c **1 / 1**

✓ **- 0 pts** Correct

QUESTION 9

9 VNA: ALU and Data Rep: d **0 / 1**

✓ **- 1 pts** Incorrect. Correct answer is "How the bytes of a multi-byte value are ordered within memory."

QUESTION 10

10 VNA: ALU and Data Rep: e **1 / 1**

✓ **- 0 pts** Correct

QUESTION 11

11 Signed and Unsigned Integers: a **1 / 1**

✓ **- 0 pts** Correct

QUESTION 12

12 Signed and Unsigned Integers: b **1 / 1**

✓ **- 0 pts** Correct

QUESTION 13

13 Signed and Unsigned Integers: c **1 / 1**

✓ **- 0 pts** Correct

QUESTION 14

14 Signed and Unsigned Integers: d **1 / 1**

✓ **- 0 pts** Correct

QUESTION 15

15 Signed and Unsigned Integers: e **1 / 1**

✓ **- 0 pts** Correct

QUESTION 16

16 Software Anatomy: a **0 / 1**

✓ **- 1 pts** Incorrect. The correct answer is "A sequence of one or more opcodes loaded in memory"

QUESTION 17

17 Software Anatomy: b **1 / 1**

✓ **- 0 pts** Correct

QUESTION 18

18 Software Anatomy: c 1 / 1

✓ - 0 pts Correct

QUESTION 19

19 Software Anatomy: d 0 / 1

✓ - 1 pts Incorrect. The answer is "To achieve the equivalent of ”if, then, else” on an Intel CPU based computer like kex we require".

QUESTION 20

20 Software Anatomy: e 1 / 1

✓ - 0 pts Correct

QUESTION 21

21 Advanced Assembly: a 0 / 1

✓ - 1 pts Incorrect

QUESTION 22

22 Advanced Assembly: b 1 / 1

✓ - 0 pts Correct

QUESTION 23

23 Advanced Assembly: c 1 / 1

✓ - 0 pts Correct

QUESTION 24

24 Advanced Assembly: d 1 / 1

✓ - 0 pts Correct

QUESTION 25

25 Advanced Assembly: e 1 / 1

✓ - 0 pts Correct

QUESTION 26

26 Part B: 6 1 / 1

✓ - 0 pts Correct

QUESTION 27

27 Part B: 7 2 / 2

✓ - 0 pts Correct

QUESTION 28

28 Part B: 8 1 / 1

✓ - 0 pts Correct

QUESTION 29

29 Part B: 9 3 / 3

✓ - 0 pts Correct

QUESTION 30

30 Part B: 10 1 / 1

✓ - 0 pts Correct

QUESTION 31

31 Part B: 11 1 / 1

✓ - 0 pts Correct

QUESTION 32

32 Part B: 12 2 / 2

✓ - 0 pts Correct

QUESTION 33

33 Part B: 13 2 / 2

✓ - 0 pts Correct

QUESTION 34

34 Part B: 14 0 / 2

✓ - 2 pts Incorrect

CS 210 Computer Systems, Spring 2020
Midterm : Lectures 1 - 8
Feb 20th, 2020

Instructions

NO ELECTRONIC DEVICES! You may use one 8.5 x 11 inch page of handwritten notes. For all multiple choice questions fill in ONE and ONLY ONE circle. Fill the circle in completely.

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 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 14 questions, for a total of 40 points.

First Name: Jeongyong Last Name: Yang

BU ID: U95912941

First Name: Jeongyong Last Name: Yang BU ID: U95912941

PART A: Multiple Choice

1. **VNA: Overview**

(a) (1 point) What are Registers?

- ○ A large collection of memory units attached to a cpu via wires we call a bus.
- ○ A set of locations within the CPU to hold values
- ○ 64 Bit values
- ● All of the above
- ○ None of the above

(b) (1 point) Which statement best characterizes how a cpu operates?

- ○ The cpu processes ascii programs.
- ○ The cpu successively adds numbers.
- ○ The cpu is a flow control device.
- ● The cpu runs a built in fetch, decode, execute loop.
- ○ None of the above

(c) (1 point) The program counter is:

- ○ a special register that counts the number of instructions executed.
- ○ typically used to track the top address of the stack.
- ○ a register used to accumulate the result of additions.
- ○ a special register that is composed of several single bit condition flags.
- ○ All of the above
- ● None of the above

(d) (1 point) Memory is a critical component of a computer because

- ○ it provides a large array in which data can be stored.
- ○ it is used to store the opcodes that form a program.
- ○ a cpu has a direct connection to memory.
- ● all of the above
- ○ none of the above

(e) (1 point) Instruction Set Architecture (ISA) refers to

- ○ a set of rules for constructing buildings.
- ● the set of registers and operations that a CPU supports.
- ○ the layout of opcodes in memory.
- ○ the way registers are broken down into sub-registers of different sizes.
- ○ all of the above.
- ○ none of the above.

First Name: Jeong Yong Last Name: Yang BU ID: U45912941

2. **VNA: ALU and Data Representation**

(a) (1 point) Software is

- ○ a sequence of bytes that encodes operations for the CPU to execute.
- ○ a set of circuits in the CPU for arithmetic and logic operations.
- ○ a file containing ascii values.
- ● all of the above.
- ○ none of the above.

(b) (1 point) Firmware is

- ○ a cpu mode in which the fetch, decode, execute loop is frozen.
- ○ a hardware mechanism that over rides all values in memory
- ● software that is already in memory and executes as soon as the processor is turned on.
- ○ all of the above.
- ○ none of the above.

(c) (1 point) The assembler

- ○ is a tool for generating binary encoded opcodes.
- ○ can process a file that contains a program written in the mnemonics of a particular CPU as input.
- ○ generates a binary object file.
- ○ is critical to making a computer programmable by humans.
- ● all of the above.
- ○ none of the above.

(d) (1 point) What is endianess?

- ○ A way of determining which end to eat an egg from.
- ○ How the bits of a single byte are order.
- ○ How the bytes of a multi-byte value are ordered within a register
- ○ How the bytes of a multi-byte value are ordered within memory.
- ● All of the above
- ○ None of the above

(e) (1 point) jmp is considered an ALU operations

- ○ True
- ● False

First Name: Jeong yong Last Name: Yang BU ID: U95912941

3. **Signed and Unsigned Integers**

(a) (1 point) A two's complement 64bit little endian computer can:

- ○ represent more signed values than unsigned values.
- ○ represent more unsigned values than signed values.
- ● can represent exactly the same number of signed values and unsigned values.

(b) (1 point) A two's complement 64bit little endian computer:

- ○ always produces the correct value when adding unsigned numbers
- ○ always produces the correct value when adding signed numbers
- ○ never produces the correct value when adding 1 to -1
- ○ all of the above
- ● none of the above

(c) (1 point) Assuming 2's complement and that T_{min} is the most negative signed value that can be represented and T_{max} is the largest positive signed value

- ○ $T_{max} + T_{min} = 0$
- ● $T_{max} + T_{min} = -1$
- ○ $T_{max} + T_{min} = 1$
- ○ $T_{max} + T_{min} = -2$
- ○ None of the above

(d) (1 point) Assuming 2's complement and a 64 bit computer which of the following are true:

- ○ 0xffffffffffffffff = -1
- ○ (bitwise complement of x) + 1 = -x
- ○ the most significant bit (bit 63) indicates the sign
- ● all of the above
- ○ none of the above

(e) (1 point) Assume rax=0x0 and rbx=0xffffffffffffffff (all 64bits are one). Then which of the following statements are true

- ○ rax is less than rbx
- ○ rax is greater than rbx
- ● it depends on which integer interpretation you use

First Name: Jeongyong Last Name: Yang BU ID: U95912941

4. **Software Anatomy**

(a) (1 point) The most basic unit of programming a computer is

- ○ A set of java statements enclosed between { and }
- ○ A sequence of one or more opcodes loaded in memory
- ○ An assembly directive
- ○ All of the above
- ● None of the above

(b) (1 point) By default the CPU will

- ○ execute add instructions before jmp instructions
- ○ execute jmp instructions before add instructions
- ○ move the value in the pc to the address specified in the stack pointer
- ● advance the pc to the address of the next instruction in linear order
- ○ add two values in memory
- ○ All of the above
- ○ None of the above

(c) (1 point) By "jumping" to an address of a prior opcode we can

- ○ perform additions more efficiently
- ● we can repeat a sequence of instructions
- ○ we can ensure that we do not run out of registers
- ○ All of the above.
- ○ None of the above

(d) (1 point) To achieve the equivalent of "if, then, else" on an Intel CPU based computer like kex we require

- ○ a high-level programming language
- ○ a stack
- ● a mov instruction and a conditional jump
- ○ an ALU instruction and a conditional jump
- ○ all of the above
- ○ none of the above

(e) (1 point) On an Intel CPU based computer eflags is

- ○ a register
- ○ composed of several single bit flags
- ○ modified by some instructions when a result is zero
- ○ is critical to implementing conditional logic
- ● All of the above
- ○ None of the above

First Name: Jeong yong Last Name: Yang BUID: U95912941

5. **Advanced Assembly language**

(a) (1 point) Which of the statements are true?

- ○ any use of jump to change the pc value is a function call
- ○ the use of a stack makes programming with functions easier
- ○ function execution requires a return address to be passed in register rdx
- ○ function execution results in the pc's value being set to a address that is larger than the current address
- ● all of the above
- ○ none of the above

(b) (1 point) On an Intel based computer the rsp register is

- ○ used to point to an area of memory that we treat as a stack.
- ○ updated by the push and pop instructions
- ○ is 64bits in length
- ● all of the above
- ○ none of the above

(c) (1 point) The 'call' instruction does the following

- ○ makes a phone call to your mother when there is a bug in your code
- ● pushes the address of the next instruction in linear order prior to setting the pc to the operand value
- ○ must immediately be followed by a compare instruction
- ○ must immediately followed by a ret instruction
- ○ none of the above

A/B

(d) (1 point) The 'ret' instruction

- ○ sets rip to the value in the rdx register
- ○ pushes the address of the next instruction in linear order prior to setting the pc
- ○ conditionally jumps if the zero flag is 1
- ○ all of the above
- ● none of the above

(e) (1 point) When we use a stack to implement function/procedure calls

- ○ we cannot nest more than 16 calls
- ○ we must carefully organize all our functions to be located at addresses that match the calling order (eg the address of the callee must be at an address that is greater than the caller)
- ● arbitrary depth recursion is possible (bounded by memory)
- ○ all of the above
- ○ none of the above

First Name: Jeongyong Last Name: Yang BU ID: U95912941

PART B : Putting it together

```
1  start:
2          movq $0x4000, %rax
3          movq $1, %rdi
4          movq $0xabcdabcdabcdabcd, %rbx
5          movq $0x8000000000000000, %rcx
6          call myfunc1
7          call myfunc2
8          call myfunc3
9          int3
10 myfunc1:
11         movq $0x1badcafefeedbeef, %r8
12         movq %r8, (%rax)
13         ret
14 myfunc2:
15         andq $-1, %rbx
16         movq %rbx, (%rax,%rdi,8)
17         ret
18 myfunc3:
19         incq %rdi
20         cmp  $0, %rcx
21         jl L1
22         movq $1, 0x4000(,%rdi,8)
23         jmp L2
24 L1:
25         incq %rdi
26         movq $-1, 0x4000(,%rdi,8)
27 L2:
28         ret
```

Assume we assemble this code and load it so 'start' is located at 0x1000 and the pc is initialized to 0x1000. We allow execution to proceed it will execute the opcodes at 'start' and proceed until it reaches the 'int3' at line 9 of the source code.

First Name: Jeongyong Last Name: Yang BU ID: U95912941

At the end of execution what will the value of the following be.

NOTE: All answers should be written as 8 byte hexadecimal numbers. You may skip insignificant zeros (leading zeros).

6. (1 point) rax : 0x4000
7. (2 points) rbx : 0xabcdabcdabcdabcd
8. (1 point) rcx : 0x8000000000000000
9. (3 points) rdi : 3
10. (1 point) r8 : 0x1badcafefeedbeef

What are the 8 byte quantities at the following memory locations. You may ignore endianness. Eg. The address indicated is the starting address of 8 bytes and you should write the value out as if it were in a register – if 0xdeadbeefdeadbeef were stored at 0x6000 then you would write down 0xdeadbeefdeadbeef (ignoring endianess).

If a location was not updated by the code the the location will hold the value 0

11. (1 point) 0x4000 : 0x1badcafefeedbeef
12. (2 points) 0x4008 : 0xabcdabcdabcdabcd
13. (2 points) 0x4010 : 0
14. (2 points) 0x4018 : -1