student name [

American University of Central Asia Software Engineering Program

Computer Architecture (COM 410)

Midterm Examination

- You have one hour and fifteen minutes to finish the test.
- You can cross answers selected by a mistake.
- You can use the back of the sheets of paper to make notes or to trace code.
- 1. What is the name of the component that executes program instructions?
 - a) CPU
 - b) Memory
 - c) Bus
 - d) Disk
- 2. How many bits are there in a byte?
 - a) 1
 - b) 2
 - c) 4
 - d) 8
- 3. Select a list of types of memory ordered correctly from fastest to slowest.
 - a) Memory / CPU registers / CPU caches
 - b) Memory / Disk drive / CPU registers
 - c) CPU registers / CPU caches / Mem-
 - d) Disk drive / Memory / CPU caches
- 4. Which sequence from the following list outlines all the major steps a compiler front end such as GCC should go through to generate a program from languages such as C or C++?
 - a) Compiling into assembly, translating into machine code
 - b) Preprocessing, compiling into assembly, translating into machine code
 - Preprocessing, compiling into assembly, translating into machine code, linking
 - d) Preprocessing, compiling into assembly, translating into machine code, linking, transforming into the bytecode
- 5. Central processing units can only execute [].
 - a) C source code
 - b) C++ source code
 - c) machine code
 - d) assembly code
- 6. The mobile Apple A11 CPU and the desktop Intel Core i7 8700K CPU have the same ISA.
 - a) Yes

- b) No
- 7. What is the MSB of the binary number 1000?
 - a) 0
 - b) 1
- 8. What is the LSB of the binary number 1000?
 - a) 0
 - b) 1
- 9. Convert the binary number 00101010 to the appropriate decimal number.
- 10. Convert the decimal number 129 to the appropriate binary number.
- 11. Convert the octal number 77 to the appropriate decimal number.
- 12. Convert the hexadecimal number FF to the appropriate decimal number. [
- 13. Convert the hexadecimal number 4A to the appropriate binary number.
- 14. According to the Linux x86-64 ABI, the result of a function should be returned in the register...
 - a) rax.
 - b) *rbx*.
 - c) *rdx*.
 - d) rcx.
 - e) rdi.
 - f) rsi.
- 15. According to the Linux x86-64 ABI, the third argument to a function should go to the register...
 - a) rax.
 - b) *rbx*.
 - c) rdx.
 - d) rcx.
 - e) rdi.
 - f) rsi.
- 16. What will be the output of the following code (write the answer)?

```
.section .data

format:
    .string "%ld\n"

.section .text
.global main
main:
    mov $42, %rsi
    lea format(%rip), %rdi
    xor %eax, %eax
    call printf@plt

    xor %eax, %eax
    ret
```

17. What will be the output of the following code?

```
.section .data

format:
    .string "%ld\n"

.section .text
.global main
main:
    mov $42, %rsi
    inc %rsi
    inc %rsi
    idec %rsi
    lea format(%rip), %rdi
    xor %eax, %eax
    call printf@plt
    xor %eax, %eax
    ret
```

18. What will be the output of the following code?

```
.section .data

format:
    .string "%ld\n"

.section .text
.global main
main:
    mov $128, %rax
    mov $128, %rsi
    add %rax, %rsi
    lea format(%rip), %rdi
    xor %eax, %eax
    call printf@plt

    xor %eax, %eax
    ret
```

19. What will be the output of the following code?

```
.section .data

format:
    .string "%d\n"
.section .text
.global main
main:
    mov $127, %al
    mov $1, %dl
    add %dl, %al
```

```
; sign extend byte to ; word in %al to %ax
                                                                                                                                                    mov b(%rip), %edx cmp %edx, %eax
            cbw
                                                                                call printf@plt
                                                                                                                                                    cmp %edx,
            cwde
                      sign extend word to
doubleword in %ax to %eax
                                                                                                                                                    ja .main.branch_a
                                                                                xor %eax, %eax
           mov %eax, %esi
lea format(%rip), %rdi
xor %eax, %eax
                                                                                                                                              .main.branch b:
                                                                         .main.branch_a:
    lea branch_a_msg(%rip), %rdi
    jmp .main.print
                                                                                                                                                    lea branch_b_msg(%rip), %rdi
            call printf@plt
                                                                                                                                              .main.print:
                                                                                                                                                    xor %eax, %eax
                                                                                                                                                     call printf@plt
            xor %eax, %eax
                                                                    23. What will be the output of the following
                                                                                                                                                     xor %eax, %eax
                                                                                                                                                     ret
20. What will be the output of the following
                                                                                                                                              .main.branch a:
                                                                          .section .data
                                                                                                                                                    lea branch_a_msg(%rip), %rdi
jmp .main.print
     code?
                                                                         .section .data
                                                                                                                                              [
                                                                         \begin{array}{c} branch\_b\_msg:\\ .string \ "branch \ b \backslash n" \end{array}
     format:
           .string "%d\n"
                                                                                                                                        26. What will be the output of the following
                                                                         a :
                                                                                                                                              code?
     .section .text
                                                                                .int −42
                                                                                                                                              .section .data
       global main
                                                                                .int -1
     main:
                                                                                                                                              \begin{array}{c} branch\_a\_msg:\\ .\ string \ "\ branch\ a \backslash n" \end{array}
           xor %ax, %ax
mov $127, %al
mov $1, %dl
add %dl, %al
                                                                         .section .text
                                                                                                                                              branch_b_msg:
.string "branch b\n"
                                                                          .global main
                                                                         main:
                                                                               mov a(%rip), %eax
mov b(%rip), %edx
cmp %edx, %eax
            adc $0,
                            %ah : add source to
           adc $0, %ah; add source to ; destination as ; with 'add' PLUS ; the carry flag cwde; sign extend word to ; doubleword in %ax to %eax
                                                                                                                                              number:
                                                                                                                                                     int 42
                                                                                jg .main.branch_a
                                                                                                                                              .section .text
                                                                         .main.branch_b:
           mov %eax, %esi
lea format(%rip), %rdi
xor %eax, %eax
                                                                                \textbf{lea} \hspace{0.1cm} \textbf{branch\_b\_msg(\%\,ri\,p)} \hspace{0.1cm}, \hspace{0.1cm} \% r\, d\, i
                                                                                                                                               .global main
                                                                                                                                              main:
                                                                                                                                                    mov number(%rip), %eax
                                                                          .main.print:
                                                                                                                                                    jp .main.branch_a ; jump if a parity ; flag was set
                                                                                xor %eax, %eax
call printf@plt
            call printf@plt
            xor %eax, %eax
                                                                                                                                              . \\ \\ \textbf{main.branch\_b:} \\ \textbf{lea} \ b \\ \\ \textbf{ranch\_b\_msg}(\% \, ri\, p \,) \;, \; \% \\ \\ \textbf{r} \, d\, i \\ \\ \\ \end{aligned}
                                                                                xor %eax, %eax
            ret
                                                                          .main.branch_a:
lea branch_a_msg(%rip), %rdi
                                                                                                                                              .main.print:
                                                                                                                                                    xor %eax, %eax
call printf@plt
21. What will be the output of the following
                                                                                jmp .main.print
                                                                                                                                                    xor %eax, %eax
     .section .data
     \begin{array}{c} branch\_a\_msg:\\ .\,string \ "\,branch\ a\backslash n" \end{array}
                                                                    24. What will be the output of the following
                                                                                                                                              .main.branch_a:
    lea branch a msg(%rip), %rdi
                                                                         code?
     branch_b_msg:
.string "branch b\n"
                                                                                                                                                    jmp .main.print
                                                                          section .data
                                                                                                                                              number:
                                                                         \begin{array}{c} branch\_a\_msg:\\ .string \ "branch \ a \backslash n" \end{array}
            .int 0
                                                                         \begin{array}{c} b \\ - b \\ - msg \\ \cdot \\ s \\ t \\ r \\ i \\ n \\ g \end{array} "b \\ r \\ a \\ r \\ b \\ \backslash n \\ " \end{array}
                                                                                                                                        27. Write the code to calculate the sum of
      .section .text
                                                                                                                                              numbers between 1 and 100.
       global main
                                                                                .int 3147483647
     main:
                                                                                                                                              The calculation should not use any for-
           mov number(\% rip), %eax
                                                                                                                                              mulas and the result should be computed
            \mathbf{je} \ . main.branch\_a
                                                                                int 1
                                                                                                                                              step by step on the CPU. (5 points)
     .main.branch b:
                                                                         .section .text
            \textbf{lea} \hspace{0.1cm} \texttt{branch\_b\_msg}(\% \hspace{0.1cm} \texttt{rip}\hspace{0.1cm}) \hspace{0.1cm}, \hspace{0.1cm} \% \texttt{rdi}
                                                                                                                                              .section .data
                                                                          .global main
                                                                                                                                              output_format:
.string "%d\n'
     .main.print:
xor %eax, %eax
                                                                               mov a(%rip), %eax
mov b(%rip), %edx
cmp %eax, %edx
            call printf@plt
                                                                                                                                              .section .text
            xor %eax, %eax
                                                                                jl .main.branch_a
                                                                                                                                               global main
            ret
                                                                          .\,main.\,branch\_\,b:
                                                                                                                                              main:
                                                                                                                                                     ; write your code on the ; back of the sheets of paper
                                                                                lea branch_b_msg(%rip), %rdi
     .main.branch_a:
    lea branch_a_msg(%rip), %rdi
jmp .main.print
                                                                          .main.print:
                                                                                xor %eax, %eax call printf@plt
                                                                                                                                                    xor %eax. %eax
22. What will be the output of the following
                                                                                xor %eax, %eax
                                                                                                                                        28. Write the code to calculate the factorial
     code?
                                                                                ret
                                                                                                                                              of 20.
                                                                          .main.branch a:
     .section .data
                                                                                lea branch_a_msg(%rip), %rdi
                                                                                                                                              The result should be computed in a
     branch_a_msg:
    .string "branch a\n"
branch_b_msg:
    .string "branch b\n"
                                                                                jmp .main.print
                                                                                                                                              separate function called factorial. (5
                                                                         ſ
                                                                                                                                              points)
                                                                    25. What will be the output of the following
                                                                                                                                              .section .data
     a :
                                                                         code?
                                                                                                                                              \begin{array}{c} \mathtt{output\_format:} \\ \mathtt{.string} \quad "\% \mathtt{lu} \backslash \mathtt{n} " \end{array}
            int 42
                                                                         .section .data
            . i n t -1
                                                                         .section .text
     .section .text
                                                                         \begin{array}{c} branch\_b\_msg:\\ .string \ "branch \ b \backslash n" \end{array}
                                                                                                                                              factorial:
                                                                                                                                                     ; write your code on the
; back of the sheets of paper
       global main
     main:
           mov a(%rip), %eax
mov b(%rip), %edx
cmp %edx, %eax
                                                                                int 42
           cmp %edx, %eax
jg .main.branch a
                                                                                .int -1
                                                                                                                                               .global main
     . \\ \\ \text{main.branch\_b:} \\ \\ \text{lea} \ \\ \text{branch\_b\_msg(\%rip)} \ , \ \% \\ \\ \text{rdi} \\
                                                                         .section .text
                                                                                                                                                    ; write your code on the ; back of the sheets of paper
                                                                          .global main
```

main:

mov a(%rip), %eax

.main.print:

xor %eax, %eax

xor %eax, %eax