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 / Memory
   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
   c) 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
    dec %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
    cbw   ; sign extend byte to
          ; word in %al to %ax
    cwde  ; sign extend word to
          ; doubleword in %ax to %eax
    mov %eax, %esi
    lea format(%rip), %rdi
    xor %eax, %eax
    call printf@plt

    xor %eax, %eax
    ret
```

[ ]

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

```
.section .data

format:
    .string "%d\n"

.section .text

.global main
main:
    xor %ax, %ax
    mov $127, %al
    mov $1,   %dl
    add %dl,  %al
    adc $0,   %ah ; add source to
                  ; destination as
                  ; with ‘add‘ PLUS
                  ; the carry flag
    cwde  ; sign extend word to
          ; doubleword in %ax to %eax
    mov %eax, %esi
    lea format(%rip), %rdi
    xor %eax, %eax
    call printf@plt

    xor %eax, %eax
    ret
```

[ ]

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

```
.section .data

branch_a_msg:
    .string "branch a\n"
branch_b_msg:
    .string "branch b\n"

number:
    .int 0

.section .text

.global main
main:
    mov number(%rip), %eax
    je .main.branch_a

.main.branch_b:
    lea branch_b_msg(%rip), %rdi

.main.print:
    xor %eax, %eax
    call printf@plt

    xor %eax, %eax
    ret

.main.branch_a:
    lea branch_a_msg(%rip), %rdi
    jmp .main.print
```

[ ]

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

```
.section .data

branch_a_msg:
    .string "branch a\n"
branch_b_msg:
    .string "branch b\n"

a:
    .int 42
b:
    .int −1

.section .text

.global main
main:
    mov a(%rip), %eax
    mov b(%rip), %edx
    cmp %edx, %eax
    jg .main.branch_a

.main.branch_b:
    lea branch_b_msg(%rip), %rdi

.main.print:
    xor %eax, %eax
    call printf@plt

    xor %eax, %eax
    ret

.main.branch_a:
    lea branch_a_msg(%rip), %rdi
    jmp .main.print
```

[ ]

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

```
.section .data

branch_a_msg:
    .string "branch a\n"
branch_b_msg:
    .string "branch b\n"

a:
    .int −42
b:
    .int −1

.section .text

.global main
main:
    mov a(%rip), %eax
    mov b(%rip), %edx
    cmp %edx, %eax
    jg .main.branch_a

.main.branch_b:
    lea branch_b_msg(%rip), %rdi

.main.print:
    xor %eax, %eax
    call printf@plt

    xor %eax, %eax
    ret

.main.branch_a:
    lea branch_a_msg(%rip), %rdi
    jmp .main.print
```

[ ]

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

```
.section .data

branch_a_msg:
    .string "branch a\n"
branch_b_msg:
    .string "branch b\n"

a:
    .int 3147483647
b:
    .int 1

.section .text

.global main
main:
    mov a(%rip), %eax
    mov b(%rip), %edx
    cmp %eax, %edx
    jl .main.branch_a

.main.branch_b:
    lea branch_b_msg(%rip), %rdi

.main.print:
    xor %eax, %eax
    call printf@plt

    xor %eax, %eax
    ret

.main.branch_a:
    lea branch_a_msg(%rip), %rdi
    jmp .main.print
```

[ ]

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

```
.section .data

branch_a_msg:
    .string "branch a\n"
branch_b_msg:
    .string "branch b\n"

a:
    .int 42
b:
    .int −1

.section .text

.global main
main:
    mov a(%rip), %eax
    mov b(%rip), %edx
    cmp %edx, %eax
    ja .main.branch_a

.main.branch_b:
    lea branch_b_msg(%rip), %rdi

.main.print:
    xor %eax, %eax
    call printf@plt

    xor %eax, %eax
    ret

.main.branch_a:
    lea branch_a_msg(%rip), %rdi
    jmp .main.print
```

[ ]

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

```
.section .data

branch_a_msg:
    .string "branch a\n"
branch_b_msg:
    .string "branch b\n"

number:
    .int 42

.section .text

.global main
main:
    mov number(%rip), %eax
    jp .main.branch_a ; jump if a parity
                      ; flag was set

.main.branch_b:
    lea branch_b_msg(%rip), %rdi

.main.print:
    xor %eax, %eax
    call printf@plt

    xor %eax, %eax
    ret

.main.branch_a:
    lea branch_a_msg(%rip), %rdi
    jmp .main.print
```

[ ]

27. Write the code to calculate the sum of numbers between 1 and 100.

The calculation should not use any formulas and the result should be computed step by step on the CPU. (5 points)

```
.section .data

output_format:
    .string "%d\n"

.section .text

.global main
main:
    ; write your code on the
    ; back of the sheets of paper

    xor %eax, %eax
    ret
```

28. Write the code to calculate the factorial of 20.

The result should be computed in a separate function called *factorial*. (5 points)

```
.section .data

output_format:
    .string "%lu\n"

.section .text

factorial:
    ; write your code on the
    ; back of the sheets of paper

    ret

.global main
main:
    ; write your code on the
    ; back of the sheets of paper

    xor %eax, %eax
    ret
```