**Computer Science Department**
**California State University, Fullerton**

CPSC 240-09 Computer Organization and Assembly Language
Quiz 03
Thursday, November 16, 2023

Student Name: _____

Last 4 digits of ID: _____

**Note:**

- University regulations on academic honesty will be strictly enforced.
- You have 75 minutes to complete this Quiz.
- You can use calculator for the Quiz.
- Close books, slides, and turn off the computer.
- Turn off or turn vibration your cell phone.
- Any content submitted after the due date will be regarded as a make-up quiz.

1. What will be in the **ax, bx,** and **dx** registers after execution? What is in **num1, num2,** and **num3** memories before execution and after execution until the label "done"? Show answer in hex, full register size. ***Note***, pay close attention to the register sizes (8-bit, 16-bit, 32-bit, or 64-bit).

```
section .data
num1  dw    60                      ;num1 = 60 = 0x003C
num2  dw    90                      ;num2 = 90 = 0x005A
num3  dw    0                       ;num3 = 0 = 0x0000

section .text
      global   _start
_start:
      mov   ax, word[num1]          ;ax = 60 = 0x003C
      mov   bx, word[num2]          ;bx = 90 = 0x005A
      mul   bx                      ;dx:ax = ax*bx = 60*90 = 5400
      mov   word[num3], ax          ;[num3] = ax = 5400 = 0x1518
done:
      mov   rax, 60
      mov   rdi, 0
      syscall
```

(26 points)

| Memory | Offset | Value (Hex) | |
|---|---|---|---|
| | | before(initial) | after |
| num3 | +1 | 0x00 | 0x15 |
| num3 | +0 | 0x00 | 0x18 |
| num2 | +1 | 0x00 | 0x00 |
| num2 | +0 | 0x5A | 0x5A |
| num1 | +1 | 0x00 | 0x00 |
| num1 | +0 | 0x3C | 0x3C |

| Register | Value (Hex) |
|---|---|
| | after execution |
| ax | 0x1518 |
| bx | 0x005A |
| dx | 0x0000 |

$5400/16 = 337\ R\ 8$

$337/16 = 21\ R\ 1$

$21/16 = 1\ R\ 5$

$1/16 = 0\ R\ 1$

$5400 = 1*16^3+5*16^2+1*16^1+8*16^0 = 0x1518$

2. What will be in the **eax, ebx, ecx,** and **edx** registers after execution. What is in **mul3** memory before execution and after execution until label "done"? Show answer in hex, full register size. ***Note***, pay close attention to the register sizes (8-bit, 16-bit, 32-bit, or 64-bit).

```
section .data
mul3    dd      0                   ;[mul3] = 0x0000 0000

section .text
        global _start
_start:
        mov     ecx, 105            ;ecx = 105
next:
        mov     edx, 0              ;edx = 0
        mov     eax, ecx            ;eax = ecx = 105, 106, 107, 108
        mov     ebx, 3              ;ebx = 3
        div     ebx                 ;edx=edx:eax % ebx = 105%3 = 0
                                    ;edx=edx:eax % ebx = 106%3 = 1
                                    ;edx=edx:eax % ebx = 107%3 = 2
                                    ;edx=edx:eax % ebx = 108%3 = 0
                                    ;eax=edx:eax / ebx = 108/3 = 36
        cmp     edx, 0              ;edx == 0
        jne     skip                ;F, T, T, F
        inc     dword[mul3]         ;[mul3]=0+1+1=2=0x00 00 00 02
skip:
        inc     ecx                 ;ecx=105+1+1+1+1=109=0x6D
        cmp     ecx, 109            ;109 == 109
        jne     next                ;T, T, T, F
done:
        mov     rax, 60
        mov     rdi, 0
        syscall
```

(24 points)

| Memory | Offset | Value (Hex) | |
|---|---|---|---|
| | | before(initial) | after |
| mul3 | +3 | 0x00 | 0x00 |
| mul3 | +2 | 0x00 | 0x00 |
| mul3 | +1 | 0x00 | 0x00 |
| mul3 | +0 | 0x00 | 0x02 |

| Register | Value (Hex) |
|---|---|
| | After execution |
| eax | 0x0000 0024 |
| ebx | 0x0000 0003 |
| ecx | 0x0000 006D |
| edx | 0x0000 0000 |

3. What will be in the **rax, rcx** and **rsi** registers after execution. What is in **len** and **array** memory before execution and after execution until the label "done"? Show memory and register answer in decimal, regardless of memory and register sizes.

```
section .data
array  db     0x64, 0x7D, 0x96, 0xAF
len    db     4

section .text
       global _start
_start:
       movzx  rcx, byte[len]         ;rcx, = 4
       mov    rsi, 0                 ;rsi = 0
pushLoop:
       mov    rax, 0                 ;rax = 0x0000 0000 0000 0000
       mov    al, byte[array+rsi]    ;al = 0x64, 0x7D, 0x96, 0xAF
       push   rax                    ;rax = 0x0000 0000 0000 00AF
       inc    rsi                    ;rsi=0+1+1+1+1
       loop   pushLoop               ;dec rcx; rcx=4-1-1-1-1
                                     ;cmp cx,0 ; jne pushLoop
       mov    rcx, qword[len]        ;rcx = 4
       mov    rsi, 0                 ;rsi = 0
popLoop:
       pop    rax                    ;rax=0x0000 0000 0000 0064
       mov    byte[array+rsi], al    ;0xAF, 0x96, 0x7D, 0x64
       inc    rsi                    ;rsi=0+1+1+1+1=4
       loop   popLoop                ;dec rcx; rcx=4-1-1-1-1=0
                                     ;cmp rcx,0 ; jne pushLoop
done:
       mov    rax, 60
       mov    rdi, 0
       syscall
```

(26 points)

| Memory | Offset | Value (Hex) | |
|---|---|---|---|
| | | before (initial) | after |
| len | +0 | 0x04 | 0x04 |
| array | +3 | 0xAF | 0x64 |
| array | +2 | 0x96 | 0x7D |
| array | +1 | 0x7D | 0x96 |
| array | +0 | 0x64 | 0xAF |

| Register | Value (Hex) |
|---|---|
| | After execution |
| rax | 0x0000 0000 0000 0064 |
| rcx | 0x0000 0000 0000 0000 |
| rsi | 0x0000 0000 0000 0004 |
| | |

| rsp | … |
|---|---|
| | 0x0000 0000 0000 0064 |
| | 0x0000 0000 0000 007D |
| | 0x0000 0000 0000 0096 |
| | 0x0000 0000 0000 00AF |
| | |
| | |

4. What will be in the **ax, bx, rcx,** and **dx** registers after execution? What is in **number** and **ascii** memory before execution and after execution until the label "done"? Show answer in hex, full register size. ***Note***, pay close attention to the register sizes (8-bit, 16-bit, 32-bit, or 64-bit).

```
section .data
number dw     987                     ;number = 987 = 0x03DB
ascii  db     "000", 0x0A

section .text
       global _start
_start:
       mov    rcx, 2                  ;rcx = 2
       mov    ax, word[number]        ;ax = 987
       mov    bx, 10                  ;bx = 10
next:
       mov    dx, 0                   ;dx = 0
       div    bx                      ;dx=dx:ax%bx=987%10=98 R 7
                                      ;dx=dx:ax%bx=98%10=9 R 8
                                      ;dx=dx:ax%bx=9%10=0 R 9
       add    byte[ascii+rcx], dl     ;ascii+2='0'+7='7'=0x37
                                      ;ascii+1='0'+8='8'=0x38
                                      ;ascii+0='0'+9='9'=0x39
       dec    rcx                     ;rcx=2-1-1-1=-1
       cmp    rcx, 0                  ;rcx < 0
       jge    next                    ;T, T, F
done:
       mov    rax, 60
       mov    rdi, 0
       syscall
```

(24 points)

| Memory | Offset | Value (Hex) | |
|---|---|---|---|
| | | before (initial) | after |
| ascii | +3 | 0x0A | 0x0A |
| ascii | +2 | 0x30 | 0x37 |
| ascii | +1 | 0x30 | 0x38 |
| ascii | +0 | 0x30 | 0x39 |
| number | +1 | 0x03 | 0x03 |
| number | +0 | 0xDB | 0xDB |

| Register | Value (Hex) |
|---|---|
| | After execution |
| ax | 0x0000 |
| bx | 0x000A |
| rcx | 0xffff ffff ffff ffff |
| dx | 0x0009 |

# ASCII TABLE

| dec | hex | oct | char | dec | hex | oct | char | dec | hex | oct | char | dec | hex | oct | char |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 000 | NULL | 32 | 20 | 040 | space | 64 | 40 | 100 | @ | 96 | 60 | 140 | ` |
| 1 | 1 | 001 | SOH | 33 | 21 | 041 | ! | 65 | 41 | 101 | A | 97 | 61 | 141 | a |
| 2 | 2 | 002 | STX | 34 | 22 | 042 | " | 66 | 42 | 102 | B | 98 | 62 | 142 | b |
| 3 | 3 | 003 | ETX | 35 | 23 | 043 | # | 67 | 43 | 103 | C | 99 | 63 | 143 | c |
| 4 | 4 | 004 | EOT | 36 | 24 | 044 | $ | 68 | 44 | 104 | D | 100 | 64 | 144 | d |
| 5 | 5 | 005 | ENQ | 37 | 25 | 045 | % | 69 | 45 | 105 | E | 101 | 65 | 145 | e |
| 6 | 6 | 006 | ACK | 38 | 26 | 046 | & | 70 | 46 | 106 | F | 102 | 66 | 146 | f |
| 7 | 7 | 007 | BEL | 39 | 27 | 047 | ' | 71 | 47 | 107 | G | 103 | 67 | 147 | g |
| 8 | 8 | 010 | BS | 40 | 28 | 050 | ( | 72 | 48 | 110 | H | 104 | 68 | 150 | h |
| 9 | 9 | 011 | TAB | 41 | 29 | 051 | ) | 73 | 49 | 111 | I | 105 | 69 | 151 | i |
| 10 | a | 012 | LF | 42 | 2a | 052 | * | 74 | 4a | 112 | J | 106 | 6a | 152 | j |
| 11 | b | 013 | VT | 43 | 2b | 053 | + | 75 | 4b | 113 | K | 107 | 6b | 153 | k |
| 12 | c | 014 | FF | 44 | 2c | 054 | , | 76 | 4c | 114 | L | 108 | 6c | 154 | l |
| 13 | d | 015 | CR | 45 | 2d | 055 | - | 77 | 4d | 115 | M | 109 | 6d | 155 | m |
| 14 | e | 016 | SO | 46 | 2e | 056 | . | 78 | 4e | 116 | N | 110 | 6e | 156 | n |
| 15 | f | 017 | SI | 47 | 2f | 057 | / | 79 | 4f | 117 | O | 111 | 6f | 157 | o |
| 16 | 10 | 020 | DLE | 48 | 30 | 060 | 0 | 80 | 50 | 120 | P | 112 | 70 | 160 | p |
| 17 | 11 | 021 | DC1 | 49 | 31 | 061 | 1 | 81 | 51 | 121 | Q | 113 | 71 | 161 | q |
| 18 | 12 | 022 | DC2 | 50 | 32 | 062 | 2 | 82 | 52 | 122 | R | 114 | 72 | 162 | r |
| 19 | 13 | 023 | DC3 | 51 | 33 | 063 | 3 | 83 | 53 | 123 | S | 115 | 73 | 163 | s |
| 20 | 14 | 024 | DC4 | 52 | 34 | 064 | 4 | 84 | 54 | 124 | T | 116 | 74 | 164 | t |
| 21 | 15 | 025 | NAK | 53 | 35 | 065 | 5 | 85 | 55 | 125 | U | 117 | 75 | 165 | u |
| 22 | 16 | 026 | SYN | 54 | 36 | 066 | 6 | 86 | 56 | 126 | V | 118 | 76 | 166 | v |
| 23 | 17 | 027 | ETB | 55 | 37 | 067 | 7 | 87 | 57 | 127 | W | 119 | 77 | 167 | w |
| 24 | 18 | 030 | CAN | 56 | 38 | 070 | 8 | 88 | 58 | 130 | X | 120 | 78 | 170 | x |
| 25 | 19 | 031 | EM | 57 | 39 | 071 | 9 | 89 | 59 | 131 | Y | 121 | 79 | 171 | y |
| 26 | 1a | 032 | SUB | 58 | 3a | 072 | : | 90 | 5a | 132 | Z | 122 | 7a | 172 | z |
| 27 | 1b | 033 | ESC | 59 | 3b | 073 | ; | 91 | 5b | 133 | [ | 123 | 7b | 173 | { |
| 28 | 1c | 034 | FS | 60 | 3c | 074 | < | 92 | 5c | 134 | \ | 124 | 7c | 174 | \| |
| 29 | 1d | 035 | GS | 61 | 3d | 075 | = | 93 | 5d | 135 | ] | 125 | 7d | 175 | } |
| 30 | 1e | 036 | RS | 62 | 3e | 076 | > | 94 | 5e | 136 | ^ | 126 | 7e | 176 | ~ |
| 31 | 1f | 037 | US | 63 | 3f | 077 | ? | 95 | 5f | 137 | _ | 127 | 7f | 177 | DEL |

www.alpharithms.com