CS-UY 2214 — Recitation 13

Introduction

Complete the following exercises. Unless otherwise specified, put your answers in a plain text file named `recitation13.txt`. Number your solution to each question. When you finish, submit your file on Gradescope. Then, in order to receive credit, you must ask your TA to check your work. Your work should be completed and checked during the recitation session.

The slides, available on Brightspace, cover many useful Intel opcodes. You may consult the online Intel assembly language reference to help you understand any unknown opcodes. You can also use the official Intel developer's manual.

Some opcodes you should know:

- `mov`: copy the value from the right operand into the left operand
- `add`, `sub`, `imul`: perform addition, subtraction, or multiplication on its two operands, storing the result in the left operand
- `xor`: performs a bitwise XOR operation on its operands, storing the result in the left operand
- `shl`, `shr`: performs a logical left or right shift on the left operand, by the number of bits given in the right operand, storing the result in the left operand
- `cmp`: compare its two operands, set the flags accordingly
- `je`, `jne`: jump if equal, and jump if not equal; to be used in combination with `cmp`

Problems

1. Consider the following fragments of Intel assembly language. What is the value of the `eax` register after executing each of the following program fragments? Give your answer as a number in hex.

(a)
```
mov eax, 0x32
```

(b)
```
mov eax, 32
```

(c)
```
xor eax, eax
```

(d)
```
mov eax, 32
mov ebx, 0x10
add eax, ebx
```

(e)
```
mov eax, 0x123456
mov ah, 0xab
```

(f)
```
mov eax, 0x123456
mov ax, 0xab
```

(g)
```
mov eax, 32
shl eax, 2
```

2. Write Intel assembly code as follows.

   (a) Write a single instruction of Intel assembly language code that will store the value 42 into register `eax`.

   (b) Write a single instruction of Intel assembly language code that will add 62 to the value in register `eax`, and store the result into `eax`.

   (c) Write a single instruction of Intel assembly language code that will add the value in register `ebx` to the value in register `eax`, and store the result into `eax`.

   (d) Write two instructions of Intel assembly language code that will add 99 to the value in register `eax`, and store the result into `ebx`.

   (e) Write two instructions of Intel assembly language code that will perform a jump to label `foo` if the value in register `eax` is equal to the value in register `ebx`, and otherwise will continue to the subsequent instruction.

   (f) Write a single instruction of Intel assembly language code that will read the 32-bit value at memory address 1099 and store it into register `eax`.

   (g) Write a single instruction of Intel assembly language code that will read the 32-bit value at label `foo` and store it into register `ecx`.

   (h) Write a single instruction of Intel assembly language code that will read the 16-bit value at address 0 and store it into register `bx`.

   (i) Write two instructions of Intel assembly language code that will read the 16-bit value at label `bar` and store it into register `eax`. Ensure that the final value of `eax` is less than 2^{16}.

   (j) Write two instructions of Intel assembly language code that will read the 32-bit value at label `foo` and store it into memory at label `bar`.

3. Consider the following fragments of Intel assembly language. What is the value of the `eax` register after executing each of the following program fragments? Give your answer as a number in hex.

   (a)
   ```
   xor eax, eax
   mov ah, 250
   add ah, 10
   ```

   (b)
   ```
   mov eax, 0aaaaaaaah
   add ah, 4
   shl eax, 4
   ```

   (c)
   ```
   mov eax, 0xaaaaaa
   mov ax, 0xffff
   xor al, al
   ```

(d)
```
xor eax, eax
mov ebx, 0xcdabff
add bl, 2
mov ax, bx
sub al, 1
```

(e)
```
mov eax, 0x5555
shr al, 1
shl eax, 16
```

4. Consider the following Intel assembly language code. Assume that execution begins at label `_start` and ends at `end`.

```
_start:
    mov eax, 0x03
    mov ebx, 4
loop:
    imul eax, eax
    sub eax, 4
    sub ebx, 1
    cmp ebx, 1
    jne loop
end:
```

What is the value of `eax` and `ebx` when execution reaches `end`? Give your answer in hex.