Anirudh Jakhotia S20190010007

COS Assignment - 2

Indian Institute of Information Technology, SriCity.

Name:- Anirudh Jakhotia
Roll No:- S20190010007

Computer Organization and Systems.

Practice Problems:-
chapter 3 : 3.1 to 3.6

Q1) Practice problem 3.1:-

Assume the following values are stored at indicated memory addresses and registers.

Address	Value	Register	Value
0x100	0xFF	%rax	0x100
0x104	0xAB	%rcx	0x1
0x108	0x13	%rdx	0x3
0x10C	0x11		

A)

operand	value.
(1) %rax	→ "0x100" (It is register operand)

Here, we are just referring the actual value of register

Anirudh Jakhotia S20190010007

(2) 0x104 → "0x AB" (Absolute address)

Here, we are referring to the value of address 0x104.

(3) \$0x108 → "0x108" (Immediate)

Here \$ indicates that we are referring to the constant value 0x108.

(4) (%rax) → "0x FF" (Address 0x100)

(%rax) refers to value at address rax, (which is 0x100) → So, the value is 0xFF.

(5) 4(%rax) → "0x AB" ("Address 0x104)

It refers to value at address rax + 4 (base + displacement) (which is 0x104). So, the value is 0x AB.

(6) 9 (%rax, %rdx) → "0x11" (Address 0x108)

It refers to value at address rax + 9 + rdx (which is 0x10C), So the value is 0x11. It is in the form Imm(rb, ri).

(7) 260 (%rcx, %rdx) → "0x13" (Address 0x108)

It refers to the value at address

Anirudh Jakhotia S20190010007

rcx + 260 + rdx [264 + 0x108 (in hexa)]

So, the value is 0x13.

$= M[260 + R(\%rcx) + R(\%rdx)]$

$= M[260 + 0x1 + 0x3]$

$= M[0x108]$

$= \underline{0x13}$

8) $\underline{0xFC(, \%rcx, 4)}$ → "$\underline{0xFF}$" (Address 0x100)

It is of form Imm(, ri, s) → [Imm + R[ri] * s]

(Scaled index). It refers to value at address (rcx * 4 + 0xFC).

Value $= M[0xFC + R(\%rcx) * 4]$

$= M[0xFC + 0x1 * 4]$

$= M[0xFC + 0x4]$

$= M[0x100] = \underline{0xFF}$

9) $\underline{(\%rax, \%rdx, 4)}$ → "$\underline{0x11}$" (Address 0x10C)

Value $= M[R(\%rax) + R(\%rdx) * 4]$

$= M[0x100 + 0x3 * 4]$

$= M[0x100 + 0xC]$

$= M[0x10C]$

$= 0x4.$

∴ So, the value is 0x11.

Anirudh Jakhotia Roll NO:- S20190010007

3.2

Q) For each of the following lines of assembly language, determine the appropriate instruction suffix based on operands.

mov - %eax, (%rsp)

mov - (%rax), (%dx)

mov - $0xFF, %bl

mov - (%rsp, %rdx, 4), %dl

mov - (%rdx), %rax

mov - %dx, (%rax)

A) We know that,

mov → b (byte)
→ w (word)
→ l (double word)
→ q (8 bytes)

Sol

(i) movl, %eax (%rsp)

→ We used 'l' to fill the blank because of %eax register beign used in the src which is a 32 bit register (or) 4 byte register

(ii) movw (%rax), %dx

→ We used 'w' to fill the blank because of the %dx register being used in

Anirudh Jakhotia S20190010007

the destination which is a 16 bit register (or) 2 byte register

(iii) mov b $ 0x FF, %bl
→ we used 'b' to fill in blank because of the %bl register which is a 1 byte register (or) 8 bit register.

(iv) mov b (%rsp, %rdx, 4), %dl.
→ We used 'b' to fill the blank because of the %dl register which is a 1 byte (or) 8 bit register

(v) mov q (%rdx), %rax.
→ we used 'q' to fill the blank are because both %rdx and %rax are 64 bit (or) 8 byte registers.

(vi) mov w (%dx, (%rax))
→ we used 'w' to fill the blank because the %dx register is a 2 byte (or) 16 bit register

Anirudh Jakhotia S20190010007

Pratice 3.3

Q3) Each of the following lines of code generates an error message when we invoke the assembler. Explain what is wrong with each line.

```
movb $0xF, (%ebx)
movl %rax, (%rsp)
movw (%rax), 4(%rsp)
movb %al, %sl
movq %rax, $0x123
movl %eax, %rdx
movb %si, 8(%rbp)
```

A) (1) movb $0xF, (%ebx)

Memory reference register must be four word. We need to change it to movb $0xF, (%rbx). We can't use %ebx as address register.

(2) movl %rax, (%rsp)

Mismatch between instruction suffix and register ID

It's four words. So we need to change it to either movl %eax, (%rsp) or movq %rax, (%rsp).

Anirudh Jakhotia S20190010007

(3) movw (%rax), 4(%rsp)

Both source and destination can't be memory references at the same time.

(4) movb %al, %sl

there is no register named as %sl

(5) movl %eax, $0x123.

$0x123 can't be destination as it is immediate value.

(6) movl %eax, %dx

Destination operand is of incorrect size. We need to change it to %edx.

(7) movb %si, 8 (%rbp)

Mismatch between instruction suffix and register ID.

%si represents a word but its given as movb [which is used for a byte].

Hence, either movb %sil, 8 (%rbp) (or) movw %si, 8(%rbp). must be the right instructions.

Q4) (Practice 3.5) (Extra-questions).

Convert the assembly code into code.

Anirudh Jakhotia S20190010007

decode 1:

```
movq (%rdi), %r8      # Get x = *xp
movq (%rsi), %rcx     # Get y = *yp.
movq (%rdx), %rax     # Get z = *zp.
movq %r8, (%rsi)      # Store x at yp.
movq %rcx, (%rdx)     # Store y at zp.
movq %rax, (%rdi)     # Store z at xp
ret
```

where xp = %rdi, yp = %rsi, zp = %rdx

A) Therefore, the req code is :-

```
void decode1 (long *xp, long *yp, long *zp)
{
    long x = *xp;
    long y = *yp;
    long z = *zp;
    *yp = x;
    *zp = y;
    *xp = z;
}
```

5) Practice Problem -3.6) :-

Suppose %rbx holds value p and %rdx holds q. Fill in the following.

Anirudh Jakhotia S20190010007

A) Given

register	value
%rbx	p
%rdx	q

→ leq instruction is a special instruction which stores the address of the source to the destination. The destination must be a register.

leaq src, dest ⇒ Dest ← &Src

(i) leaq 9(%rdx), %rax

9(%rdx) = 9 + %rdx = 9 + q = %rax

∴ Result = 9+q

(ii) leaq (%rdx, %rbx), %rax

(%rdx, %rbx) = %rdx + %rbx

= p+q = %rax

∴ Result = p+q.

(iii) leaq (%rdx, %rbx, 3), %rax.

(%rdx, %rbx, 3) = %rdx + 3 * %rbx

= q + 3p = %rax

∴ Result = q+3p

Anirudh Jakhotia S20190010007

(iv) leaq 2 (%rbx, %rbx, 7), %rax

2 (%rbx, %rbx, 7) = 2 + %rbx + 7 * %rbx

= 2 + p + 7p

= 2 + 8p

∴ Result = 2+8p.

(v) leaq 0xE (, %rdx, 3), %rax.

0xE (, %rdx, 3) = 0xE + %rdx * 3

= 14 + 3q.

∴ Result = 14 + 3q.

(vi) leaq 6 (%rbx, %rdx, 7), %rax

6 (%rbx, %rdx, 7) = 6 + %rbx + 7 * rdx

= 6 + p + 7q.

∴ Result = 6+p+7q.