Anirudh Jakhotia S20190010007

COS Assignment -3

Indian Institute of Information Technology, Sricity.

Name:- Anirudh Jakhotia

Roll No:- S20190010007

Computer Organization and Systems

Practice Problems :-

chapter 3 : 3.6 to 3.12.

Q1) Practice problem 3.7 :-

Consider the following code, in which we have omitted the expression being computed:

```
short scale3(short x, short y, short z)
{
    short t = ____;
    return t;
}
```

Anirudh Jakhotia S20190010007

Compiling the actual f^n with GCC yields the following assembly code:

short scale3 (short x, short y, short z)

x in %rdi, y in %rsi, z in %rdx.

```
scale3:
    leaq (%rsi, %rsi, 9), %rbx    → ①
    leaq (%rbx, %rdx), %rbx       → ②
    leaq (%rbx, %rdi, %rsi), %rbx → ③
    ret
```

Fill the missing expression in C code.

A) Given assembly code:

Register	value
%rdi	x
%rsi	y
%rdx	z

%rbx = t.

From instruction ①.

$\%rbx = t = (\%rsi + \%rsi * 9)$

$= y + 9*y = 10y.$

From instruction ②.

$\%rbx = t = (\%rbx + \%rdx)$

$= 10y + z.$

Anirudh Jakhotia S20190010007

From (3) instruction

%rbx = t = (%rbx + %rsi + %rdi)

= 10y + z + xy.

∴ The function:

```
short scale3(short x, short y, short z)
{ short t = 10y + z + xy;
  return t;
}
```

Q2) Practice problem 3.8:

Assume the following values and stored at indicated memory address and registers:

Address	Value	Register	value
0x100	0xFF	%rax	0x100
0x108	0xAB	%rcx	0x1
0x110	0x13	%rdx	0x3.
0x118	0x11		

Fill in the following :-

Anirudh Jakhotia   CS20190010007

A)

Instruction	Destination	Value
addq %rcx, %rax	0x100	0x100
subq %rdx, 8(%rax)	0x108	0xA8
imulq $16, (%rax, %rdx, 8)	0x118	0x110
incq 16 (%rax)	0x110	0x14
decq %rcx	%rcx	0x0
subq %rdx, %rax	%rax	0xFD

We know that

8 (%rax) → %rax + 8

(%rax, %rdx, 8) → %rax + 8 * %rdx = 0x118.

the value stored at 0x118 is 0x11

imulq multiplies by $16 (0x10 in hexa)

So, the value becomes 0x110.

① For addq it takes value of (%rax) and adds %rcx to it, So, 0xff + 0x1 = 0x100

② subq S, D : subtracts source from destination

%rdx = S, 8 (%rax) = D value = 0xAB

= 0x3.

③ imulq S, D : multiply destination by source

(%rax, %rdx, 8) = D   S = $16

value = 0x11 * 0x10 = 0x110.

(value at 0x118) ($16)

Aniruddh Jakhotia S20190010007

(4) incq 16(%rax):

We know that incq D: is unary increment of D by 1.

So D = 16(%rax) = %rax + 16 = 0x110.

[Value at 0x110] + 1 = 0x13 + 1 = 0x14

(5) decq %rcx:

we know that decq D: is unary operand that decrements D by 1.

$\boxed{D = \%rcx} \Rightarrow$ (value at %rcx) − 1.

= 0x0 = value.

(6) subq %rdx, %rax:

We know that subq S, D: Subtracts D from S.

D = $\boxed{\%rax}$ ⇒ (value at %rax) − (value at %rdx)

= 0x100 − 0x3 = $\boxed{0xFD}$.

Practice Problems 3.9:

Q3) Suppose we want to generate assembly code for the following C function:

```
long shift_left4_rightn(long x, long n)
{   x <<= 4;
    x >>= n;
    return x;
}
```

Anirudh Jakhotia S20190010007

Fill the below assembly code.

Let x = %rdi, n = %rsi

A) Shift_left4_rightn:

movq %rdi, %rax → Get x

salq $4, %rax 'x <<= 4' (Shift left op)

movl %esi, %ecx → we will access 4 bytes from n

sarq %cl, %rax → x >>= n (arithmetic right shift op)

Thus, this is the required assembly code for the given question.

(Extra questions)

Q4) Practice problem 3.10.

Assembly code is given as,

(Short arithmetic 3 (Short x, Short y, Short z)

x in %rdi, y in %rsi, z in %rdx.

arith 3:

orq %rsi, %rdx # (or operator rdx/rsi = y|z)

sarq $9, %rdx # (right shift rdx >> 9 = p1 >> 9)

Anirudh Jakhotia Roll No:- S20190010007

notq %rdx # (not operator P3 = ~P2)

movq %rdx, %rax # move operator

rax = %rdx.

subq %rsi, %rbx # sub operator

%rbx = %rbx - %rsi

⇒ (P4 = y - P3)

ret.

Based on assembly code, fill in the missing portions of C-code.

A) The req code C-code will be.

short arith3 (short x, short y, short z)

{

Short P1 = "y|z";

Short P2 = "P1>>9";

Short P3 = "~P2";

Short P4 = "y-P3";

~~Short~~ return P4;

}

Q5) Problem 3.11 :

It is common to find assembly-code lines of the form

xorq %rcx, %rcx

Anirudh Jakhotia S20190010007

in code that was generated from C where no Exclusive-or operations were present.

(A) Explain the effect of this particular Exclusive-or and what useful operations it implements

Ans) This instruction is used to set register %rcx to zero, exploiting the property x^x = 0, for any x. It corresponds to C statement x=0. In this way it helps to initialise value to zero.

(B) What would be the more straight forward way to express this operation in assembly code?

A) A more direct way of setting register %rcx, to zero it with the instruction

movq $0, %rcx.

Anirudh Jakhotia S20190010007

(C) Compare the number of bytes to encode any two of these three implementations of same operation.

A) Assembling and dissassembling this code, however, we find that the version with xorq requires only 3 bytes, while the version with movq requires 7 bytes.

the other ways to set %rcx to zero rely on the property that any instruction that updates the lower 4 bytes will cause high-order bytes to set to zero. thus, we could use either ~~xorl~~ xorl %ecx, %ex (2 bytes) (or) movl $0, %ecx (5 bytes).