Computer Architecture - 2016 Fall
Handout Assignment #1
Due date : 2016. 10. 5

(기본적인 레지스터 저장방식과 형식 이해)

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

Address	**Value**	**Register**	**Value**
0x300	0xA	%rax	0x300
0x304	0xB	%rbx	0x304
0x308	0xC	%rcx	0x1
0x30C	0xD	%rdx	0x20B

Fill in the following table

Operand	**Value**
%rdx	
0x30C	
$0x300	
(%rbx)	
12(%rax)	
11(%rax,%rcx)	
0x1(%rcx,%rcx, 302)	
0x2FA(,%rcx,6)	

(간단한 어셈블리어 코드 분석 후 의도 파악하기)

2. Consider following assembly code generated by compiler,

long func(long x, long y, long z)
x in %rdi, y in %rsi, z in %rdx

```
1       func:
2       .L0:
3               movq    %rdi, %rax
4               cmpq    %rsi, %rdi
5               jge     .L2
6               movq    %rsi, %rax
7       .L2:
8               cmpq    %rdx, %rax
9               jge     .L3
10              movq    %rdx, %rax
11      .L3:
12              ret
```

Q1. Briefly describe what does this function mainly do

Q2. If [x = 5], [y = 3], [z = 10], what would this function return in the end?

(Cooking 인스트럭션들 사용법 이해와 목적지 주소, 레지스터 그리고 그 목적지에 입력되는 값 이해)
3. Assume the following values are stored at the indicated memory address and register:

Address	**Value**	**Register**	**Value**
0x200	0xFFF	%rax	0x200
0x208	0xAB	%rcx	0x1
0x210	0x13	%rdx	0x3
0x218	0x11		

Fill in the following table showing the effects of the following instruction in terms of both the register or memory location that will be updated and the resulting value:

Instruction	**Destination**	**Value**
addq %rcx, (%rax)		
subq %rdx, 16(%rax)		
imulq $200, (%rax, %rdx, 8)		
decq %rdx		
subq %rdx, %rax		

(Assembly 코드를 분석하여 C코드 완성시키기, conditional branch 명령어와 arithmetic 명령어의 이해)
4. Consider following C code,

```
1       long func1(long x, long y, long z) {
2               long val = _________________;
3               if(_________________) {
4                       if(_________________) {
5                               val = _________________;
6                       else
7                               val =_________________;
8                       }
9               }
10              else if (_________________)
11                      if(_________________)
12                              val = _________________;
13                      else
14                              val= _________________;
15
16              return val;
17      }
```

After compiling C code above, gcc generated following assembly code:
long func1 (long x, long y, long z)
x in %rdi, y in %rsi, z in %rdx

```
1       func1:
2               movq %rsi, %rax
3               addq %rdx, %rax
4               subq %rdi, %rax
5               cmpq $10, %rdx
6               jle .L3
7               cmpq $2, %rsi
8               jle .L2
9               movq %rdi, %rax
10              idivq %rdx, %rax
11              ret
12      .L2:
13              movq %rdi, %rax
14              idivq %rsi, %rax
15              ret
16      .L3:
17              cmpq $6, %rdx
18              jge .L4
19              cmpq $2, %rsi
20              jle .L2
21              movq %rdx, %rax
22              subq %rsi, %rax
23      .L4:
24              ret
```

(C언어의 while 루프를 assembly 언어로 구현하기, conditional branch operation의 이해)

5. Consider the following C code,

```
1       long while_in_asm(long a, long b){
2               long  result = 0;
3               while (a > b) {
4                       result = result + (a*b);
5                       a = a-1;
6               }
7               return result;
8       }
```

Fill out the generated assembly code that is equivalent to C code above.

long while_in_asm(long a, long b)
a in %rdi, b in %rsi

return value must be stored at %rax before return

```
1       while_in_asm:
2               movq $0, _________                  ;to initialize result to zero
3               jmp     _________                  ;jump to conditional statement
4
5       .L3
6               movq        %___________, %rdx     ;store a to register
7               imulq       %rsi, ___________      ;multiply two variables
8               _______     %rdx, %rax             ;add calculation to the result
9               subq        _________,_________    ;decrement a by 1
10      .L2
11              cmpq        _________,_________    ;check if loop must stop or continue
12              jg          _________              ;jump back to calculation part
13              ret
```

(C코드를 보고 똑같은 기능을 하는 어셈블리로 바꾸기)

6. Starting with C code of the form

```
1       void branch (long a, long *p){
2               if (a == 0)
3                       goto done;
4               if(a >= *p)
5                       goto done;
6               *p = a;
7       done:
8               return;
9       }
```

*void branch(long a, long *p)*
a in %rdi, p in %rsi

Write assembly code for branch that will have an effect equivalent to the C code above.

(assembly 언어로 구현된 함수의 기능 해석하기)

7. Consider given assembly code

long rec(unsigned long x)
x in %rdi

```
1       rec:
2               pushq   %rbx
3               movq    %rdi,   %rbx
4               movq    $0,     %rax
5               testq   %rdi,   %rdi
6               je .L2
7               shrq    $2,     %rdi
8               call    rec
9               addq    %rbx,   $rax
10      .L2:
11              popq    %rbx
12              ret
```

A. What value does rec store in the callee-saved register $rbx?

B. What does this code do ?

(어셈블리어를 보고 똑같은 기능을 하는 C코드로 바꾸기)

8. Consider following assembly code generated by compiler,

long decode2(long x, long y, long z);
x in %rdi, y in %rsi and z in %rdx

```
1       decode2:
2               subq    %rdx,   %rsi
3               imulq   %rsi,   %rdi
4               movq    %rsi,   %rax
5               salq    $63,    %rax
6               sarq    $63,    %rax
7               xorq    %rdi,   %rax
8               ret
```

Q. Write C code for decode2 that will have an effect equivalent to the assembly code shown.