CSED211 Homework #2, Answer

1. Investigate the following things.
   A. What is 'Instruction Set Architecture' (ISA)?
      Check Wikipedia (https://en.wikipedia.org/wiki/Instruction_set_architecture)
   B. What is 'RISC' architecture? What are the big difference compared to 'CISC'?
      Check the following article
      (https://www.listdifferences.com/risc-and-cisc-architecture-their-differences/)
2. Read carefully section 3.6.6 and answer the following questions
   A. Why is the conditional move advantageous?
      테스트와 조건 브랜치를 없앨 수 있다. 테스트와 조건 브랜치는 다음 인스트럭션을 가져오는데 지연을 가져온다. 예측을 통하여 다음 인스트럭션을 가져온다고 하여도 예측이 틀리면 예측으로 가져온 인스트럭션의 결과를 모두 버리고 새로 맞는 곳에서 인스트럭션을 가져와야 한다. 파이프라인을 이용하여 시스템의 성능을 가져올 경우 예측이 틀림으로서 가지는 페날티는 더욱 커진다. Conditional move 는 이러한 것들 것 필요로 하지 않고 조건이 안 맞으면 move를 하지 안하도록 하여 동일한 효과를 가지게 만든다.
   B. As an invalid use of conditional move, it has shown an example of the following code (page 254, 3nd ed.)

```
long cread (long *xp) {
    return ( xp ?   *xp   :   0 );
}        // if xp is not null pointer, then return the value by dereferencing
```

// if a compiler generates an assembly code like below, it may make a problem during runtime. (Definitely, gcc compiler will not generate an assembly code like below.)

```
cread:
    movq    (%rdi), %rax
    testq    %rdi, %rdi
    movl    $0, %edx
    cmove   %rdx, %rax
    ret
```

Explain what is wrong in this example.

포인터 값이 %rdi 로 전달되었는데, 이것이 null 인 경우에 첫번째 인스트럭션인 movq (%rdi), %rax 이 메모리를 참조하고 이 것이 프로그램 에러를 발생시킨다.

3. Exercise 3.60 on page 348
   A. result = %rax. Parameter x in %rdi. Parameter n in %esi and then copied into %ecx. Register %edx is initialized to 1. Infer that mask be %rdx.
   B. They are initialized to 0 and 1, respectively.
   C. The test condition is that mask is nonzero.
   D. The salq instruction updates mask to be mask << n.
   E. Variable result is updated to be result | (x&mask).
   F. Omitted.

4. Exercise 3.63 on page 350

```
long switch_prob (long x, long n) {
    long result = x;
    switch(n) {
        case 60:
        case 62:
            result <<= 3;
            break;
        case 63:
            result >>= 3;
            break;
        case 64:
            result *= 15;
        /* Fall through */
        case 65:
            result *= result;
        /* Fall through */
        default:
            result += 75;
    }
    return result;
}
```

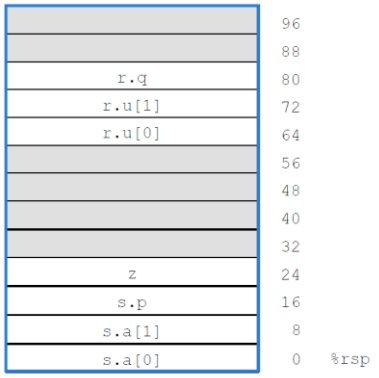
5. Exercise 3.66 on page 353

```
#define NR(n)   (3 * (n))
```

#define NC(n) (4 * (n) + 1)

6. Exercise 3.67 on page 354

A. Function 'eval' passes s to process, using 24 bytes at the top of the stack. It also stores argument z (in register %rdx) on the stack at offset 24.



B. Function 'eval' allocates 24 bytes on the stack at offset 64 for the result structure r. It passes a pointer to this region as a hidden argument to process in register %rdi.

C. Function 'process' accesses the fields of argument structure s on the stack. Since the callq instruction pushed an 8-byte return address, the fields of s start at offset 8.

D. Function 'process' sets the fields of the result structure via the pointer passed as a hidden argument.

E. See the above diagram.

F. Structure arguments are passed on the stack. When calling a function that will return a structure, the caller allocates space on its stack and passes a pointer to this region as a hidden argument to the function.

7. Exercise 3.69 on page 357

A. CNT is 7.

B.

```
#define CNT 7
typedef struct {
    long idx;
    long x[4];
} a_struct;
```