Q 3.64
R, S, T는 #define으로 선언된 상수들
long A[R][S][T];

```
long store_ele(long i, long j, long k, long *dest){
    *dest=A[i][j][k];
    return sizeof(A);
}

// i in %rdi, j in %rsi, k in %rdx, dest in %rcx
store_ele:
  leaq    (%rsi, %rsi, 2), %rax     ;%rax=3*j
  leaq    (%rsi, %rax, 4), %rax     :%rax=13*j
  movq    %rdi, %rsi                ; %rsi=i
  salq    $6, %rsi                  ; %rsi=i<<6=i*64
  addq    %rsi, %rdi                ; i=i*64+i
  addq    %rax, %rdi                ; %rdi=65*i+13*j
  addq    %rdi, %rdx                ; %rdx = 65*i+13*j+k
  movq    A(, %rdx, 8), %rax        ; A(배열의 시작 주소)+%rdx*8(sizeof)
  movq    %rax, (%rcx)              ; *dest=%rax
  movl    $3640, %eax               ; return 3640
  ret
```

A. 수식 3.1을 2차원에서 3차원으로 확장하여 배열 A[i][j][k]의 위치를 계산하도록 하라.
T D[R][C];
&D[i][j]=Xd + L(C*i + j)
//Xd는 시작 배열의 시작 주소를 나타내고, L은 배열의 사이즈를 나타낸다.
=>
TYPE D[R][S][T];
&D[i][j][k]=Xd+L(S*T*i + T*j + k)

B. 이 어셈블리 코드를 활용하여 역엔지니어링해서 R, S, T의 값을 결정하시오.
S*T = 65
T=13
=> S = 5
3640/8=455 = 13 * 5 * 7
=> R=7
따라서 R은 7, S는 5, T는 13을 나타낸다.

Q 3.67
구조체를 인자와 리턴 값으로 갖는 함수를 번역한 코드를 살펴보고 이들이 언어의 특징을 사용해서 어떻게 구현되는지 알아봄
함수 process, process를 출력하는 함수 eval

```
typedef struct{
    long a[2];
    lond *p;
} strA
typedef struct{
    long u[2];
    long q;
} strB;
strB process(strA s){
    strB r;
    r.u[0]=s.a[1];
    r.u[1]=s.a[0];
    r.q=*s.p;
    return r;
}
long eval(long, x, long y, long z){
    strA s;
    s.a[0]=x;
    s.a[1]=y;
    s.p=&z;
    strB r = process(a);
    return r.u[0] + r.u[1] + r.p;
}
```

gcc는 이 두 개의 함수에 대해 코드를 생성함

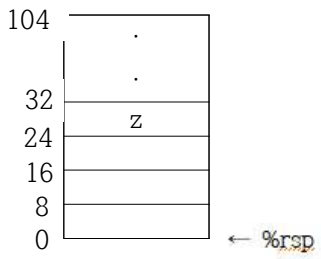
```
process:
  movq    %rdi, %rax        ;%rax=%rdi
  movq    24(%rsp), %rdx
  movq    (%rdx), %rdx
  movq    16(%rsp), %rcx
  movq    %rcx, (%rdi)
  movq    8(%rsp), %rcx
  movq    %rcx, 8(%rdi)
  movq    %rdx, 16(%rdi)
  ret
```

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

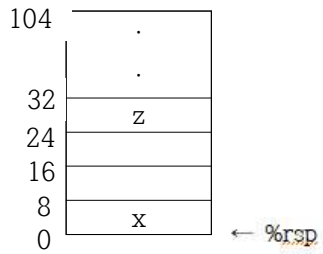
```
eval:
   subq     $104, %rsp
   movq     %rdx, 24(%rsp)
   leaq     24(%rsp), %rax
   movq     %rdi, (%rsp)
   movq     %rsi, 8(%rsp)
   movq     %rax, 16(%rsp)
   leaq     64(%rsp), %rdi
   call     process
   movq     72(%rsp), %rax
   addq     64(%rsp), %rax
   addq     80(%rsp), %rax
   addq     $104, %rsp
   ret
```

A. 2번 줄에서 함수 eval이 104바이트를 스택에 할당하는 것을 볼 수 있다. eval에 대한 스택 프레임을 그림으로 그리고, process를 호출하기 전에 스택에 저장된 값들을 표시하시오

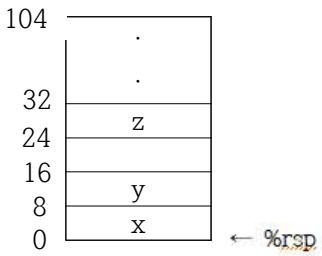
- movq %rdx,24(%rsp)




- movq %rdi, (%rsp)

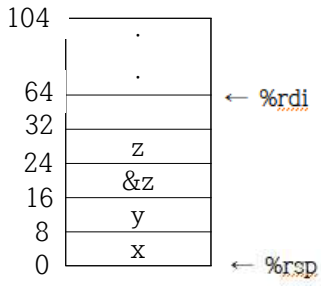



- movq %rsi, 8(%rsp)




- movq %rax, 16(%rsp) , leaq 64(%rsp), %rdi

%rax = &z




B. process를 호출할 때 eval은 어떤 값을 전달하는가?

- eval은 새로 처리할 주소 %rsp+64를 전달한다.

C. process 코드가 어떻게 구조체 인자 s의 원소들을 접근하는가?

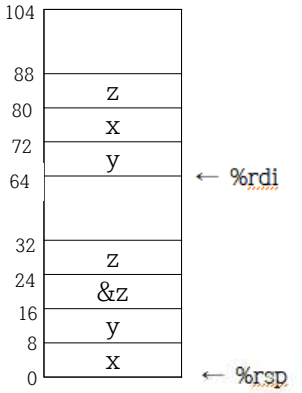
- process는 스택 포인터 %rsp에 offset을 더함으로써 접근한다.

D. process 코드가 어떻게 결과 값 구조체 r의 필드들을 설정하는가?

- eval이 주소%rsp+64를 process에 전달하면 process는 처음부터 이 주소에 데이터를 저장하고 마지막으로 이 주소를 다시 리턴한다.

E. 어떻게 eval이 process로부터 리턴한 후에 구조체 r의 원소들을 접근하는지 나타내도록 eval을 위한 스택 프레임의 그림을 완성하시오.

- movq %rdi, %rax ;%rax = %rdi (%rdi return)
  movq 24(%rsp), %rdx ;%rdx=*(%rsp+24)=&z
  movq (%rdx),%rdx ; %rdx=*(&z)=z
  movq 16(%rsp), %rcx ; %rcx = *(%rsp+16)=y
  movq %rcx, (%rdi) ; *(%rdi) = %rcx=y
  movq 8(%rsp), %rcx ; %rcx = *(%rsp+8)=x
  movq %rcx, 8(%rdi) ; *(%rdi+8)=%rcx=x
  movq %rdx, 16(%rdi) ; *(%rdi+16)=%rdx=z




F. 어떻게 구조체 값들이 함수 인자로 전달되는지, 어떻게 이들이 함수의 결과 값으로 리턴되는지에 대하여 어떤 일반적인 원칙을 알아낼 수 있는가?

- caller는 공간을 찾고 callee에게 이 공간 주소를 전달하고, callee는 이 공간에서 데이터를 저장하고 이 주소를 다시 반환한다.

Q 3.69

```
typedef struct {
    int first;
    a_struct a[CNT];
    int last;
} b_struct;
void test(long i, b_struct *bp) {
    int n= bp->first + bp->last;
    a_struct *ap=&bp->a[i];
    ap ->x[ap->idx] = n;
}
```

.o파일을 역어셈블한 결과

```
;i in %rdi, bp in %rsi
0000000000000000 <test> :
   0:   8b 8e 20 01 00 00     mov    0x120(%rsi), %ecx       ;%ecx=*(bp+0x120(288))
   6:   03 0e                 add    (%rsi), %ecx            ;%ecx=*(bp+0x120)+(*bp)
   8:   48 8d 04 bf           lea    (%rdi, %rdi, 4), %rax   ;%rax=5*i
   c:   48 8d 04 c6           lea    (%rsi, %rax, 8), %rax   ;%rax=40*i+bp
  10:   48 8b 50 08           mov    0x8(%rax), %rdx         ;%rdx=40*i+bp+8
  14:   48 63 c9              movslq %ecx, %rcx              ;%rcx=(%ecx<<32)>>32
  17:   48 89 4c d0 10        mov    %rcx, 0x10(%rax, %rdx, 8)
  ;*(((40*i+bp+8)*8)+40*i+bp+16)=%rcx
  1c:   c3                    retq
```

A. CNT 값

ap = &bp->a[i] => 40*i+bp+8에서 bp+8인 것을 보고 first변수가 차지하는 크기는 8인 것을 알 수 있다. 또한 a_struct의 크기는 40인 것을 알 수 있다.
그 다음으로 맨 처음과 둘째 줄에서 n을 구하는 식인 것을 알 수 있는데 bp->first변수는 *bp이기 때문에 bp-> last는 bp+288이다. 따라서 위에 따르면 a_struct의 크기가 40인 것을 알 수 있기 때문에 8+40*CNT는 288이다. 이 식을 풀어 CNT를 구해보면 280/40=7로
CNT의 값은 7인 것을 알 수 있다.

B. 구조체 a_struct의 완전한 선언문, 이 구조체의 필드는 오직 idx, x뿐이며 이들 모두는 부호형 값을 가진다고 가정

마지막 mov 명령문에서 %rdx에 *8을 하므로 idx는 long타입이다.
그리고 16을 더하는 것으로 보아 x배열로 long타입인 것을 알 수 있는데 아까 A의 답변에서 a_struct의 크기 40이라고 했으니 (40-8)/8=4이므로 x배열의 크기가 4인 것을 알 수 있다.
따라서 a_struct의 구조를 이렇게 예측할 수 있다.

```
typedef struct{
    long idx;
    long x[4];   }a_struct;
```