c파일

```c
#include<stdio.h>

int a = 10;

typedef struct struc{
        int b;
        int c;
        char d;
        float e;
}Struc;

int add(int x);
float sub(float x);
int product(int x[]);
float devide(Struc st);
void swap1(int x,int y);
void swap2(int *x,int *y);

int main(){
        int q = 5;
        int w = 15;
        float e = 1.5;
        int arr[2] = {20,40};
        Struc ss;
        ss.b = 7;
        ss.c = 8;
        ss.d = 't';
        ss.e = 0.5;
        int Add = add(q);
        float Sub = sub(e);
        int Pro = product(arr);
        float Dev = devide(ss);
        swap1(q,w);
        swap2(&q,&w);
}

int add(int x){
        int sum;
        int k = 11;
        sum = a+k+x;
        return sum;
}
float sub(float x){
        float sub;
        float k = 10.1;
        sub = k-x;
        return sub;
}
int  product(int x[]){
        int pro;
        pro = x[0]*x[1];
        return pro;
}
float devide(Struc st){
        float dev;
        float k = 2.5;
        dev = k / st.e;
        return dev;
}
void swap1(int x,int y){
        int k = x;
        x = y;
        y = k;
}
void swap2(int *x,int *y){
        int k = *x;
        *x = *y;
        *y = k;
}
```

어셈블리코드

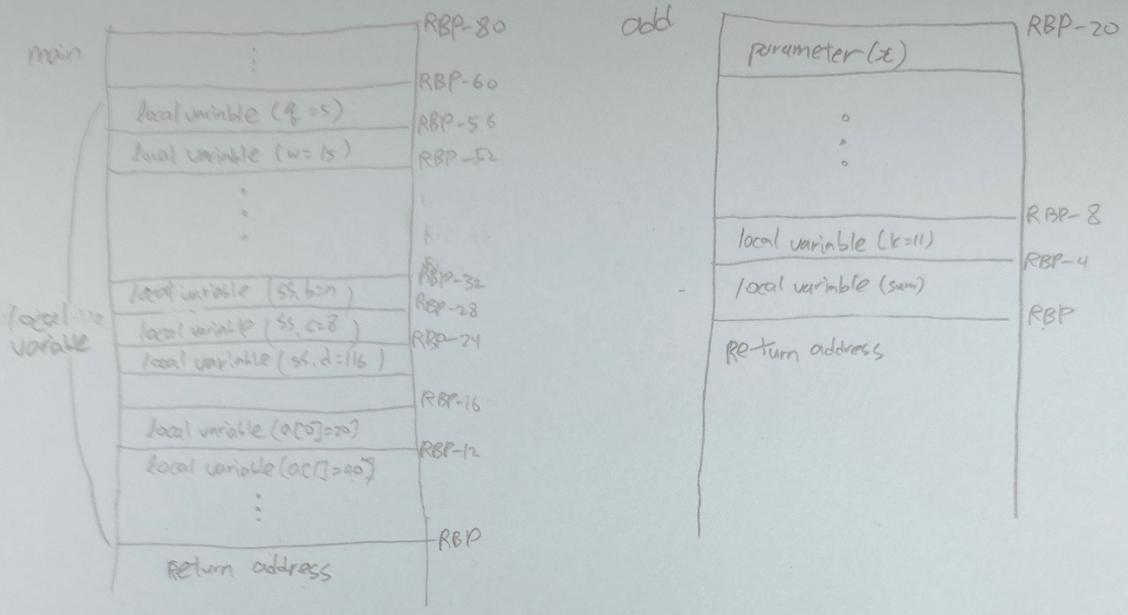
```
        .file   "hw7.c"
        .globl  a
        .data
        .align 4
        .type   a, @object
        .size   a, 4
a:
        .long   10
        .text
        .globl  main
        .type   main, @function
main:
.LFB0:
        .cfi_startproc
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        subq    $80, %rsp
        movq    %fs:40, %rax
        movq    %rax, -8(%rbp)
        xorl    %eax, %eax
        movl    $5, -60(%rbp)
        movl    $15, -56(%rbp)
        movss   .LC0(%rip), %xmm0
        movss   %xmm0, -52(%rbp)
        movl    $20, -16(%rbp)
        movl    $40, -12(%rbp)
        movl    $7, -32(%rbp)
        movl    $8, -28(%rbp)
        movb    $116, -24(%rbp)
        movss   .LC1(%rip), %xmm0
        movss   %xmm0, -20(%rbp)
        movl    -60(%rbp), %eax
        movl    %eax, %edi
        call    add
        movl    %eax, -48(%rbp)
        movl    -52(%rbp), %eax
        movl    %eax, -68(%rbp)
        movss   -68(%rbp), %xmm0
        call    sub
        movd    %xmm0, %eax
        movl    %eax, -44(%rbp)
        leaq    -16(%rbp), %rax
        movq    %rax, %rdi
        call    product
        movl    %eax, -40(%rbp)
        movq    -32(%rbp), %rdx
        movq    -24(%rbp), %rax
        movq    %rdx, %rdi
        movq    %rax, %rsi
        call    devide
        movd    %xmm0, %eax
        movl    %eax, -36(%rbp)
        movl    -56(%rbp), %edx
        movl    -60(%rbp), %eax
        movl    %edx, %esi
        movl    %eax, %edi
        call    swap1
        leaq    -56(%rbp), %rdx
        leaq    -60(%rbp), %rax
        movq    %rdx, %rsi
        movq    %rax, %rdi
        call    swap2
        movl    $0, %eax
        movq    -8(%rbp), %rcx
        xorq    %fs:40, %rcx
        je      .L3
        call    __stack_chk_fail
.L3:
        leave
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE0:
        .size   main, .-main
        .globl  add
        .type   add, @function
add:
.LFB1:
        .cfi_startproc
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        movl    %edi, -20(%rbp)
        movl    $11, -8(%rbp)
        movl    a(%rip), %edx
        movl    -8(%rbp), %eax
        addl    %eax, %edx
        movl    -20(%rbp), %eax
        addl    %edx, %eax
        movl    %eax, -4(%rbp)
        movl    -4(%rbp), %eax
        popq    %rbp
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE1:
        .size   add, .-add
        .globl  sub
        .type   sub, @function
```

```
sub:
.LFB2:
        .cfi_startproc
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        movss   %xmm0, -20(%rbp)
        movss   .LC2(%rip), %xmm0
        movss   %xmm0, -8(%rbp)
        movss   -8(%rbp), %xmm0
        subss   -20(%rbp), %xmm0
        movss   %xmm0, -4(%rbp)
        movss   -4(%rbp), %xmm0
        popq    %rbp
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE2:
        .size   sub, .-sub
        .globl  product
        .type   product, @function
product:
.LFB3:
        .cfi_startproc
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        movq    %rdi, -24(%rbp)
        movq    -24(%rbp), %rax
        movl    (%rax), %edx
        movq    -24(%rbp), %rax
        addq    $4, %rax
        movl    (%rax), %eax
        imull   %edx, %eax
        movl    %eax, -4(%rbp)
        movl    -4(%rbp), %eax
        popq    %rbp
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE3:
        .size   product, .-product
        .globl  devide
        .type   devide, @function
devide:
.LFB4:
        .cfi_startproc
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        movq    %rdi, %rax
        movq    %rsi, %rcx
        movq    %rcx, %rdx
        movq    %rax, -32(%rbp)
        movq    %rdx, -24(%rbp)
        movss   .LC3(%rip), %xmm0
        movss   %xmm0, -8(%rbp)
        movss   -20(%rbp), %xmm1
        movss   -8(%rbp), %xmm0
        divss   %xmm1, %xmm0
        movss   %xmm0, -4(%rbp)
        movss   -4(%rbp), %xmm0
        popq    %rbp
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE4:
        .size   devide, .-devide
        .globl  swap1
        .type   swap1, @function
swap1:
.LFB5:
        .cfi_startproc
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        movl    %edi, -20(%rbp)
        movl    %esi, -24(%rbp)
        movl    -20(%rbp), %eax
        movl    %eax, -4(%rbp)
        movl    -24(%rbp), %eax
        movl    %eax, -20(%rbp)
        movl    -4(%rbp), %eax
        movl    %eax, -24(%rbp)
        nop
        popq    %rbp
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE5:
        .size   swap1, .-swap1
        .globl  swap2
        .type   swap2, @function
```

```
swap2:
.LFB6:
        .cfi_startproc
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        movq    %rdi, -24(%rbp)
        movq    %rsi, -32(%rbp)
        movq    -24(%rbp), %rax
        movl    (%rax), %eax
        movl    %eax, -4(%rbp)
        movq    -32(%rbp), %rax
        movl    (%rax), %edx
        movq    -24(%rbp), %rax
        movl    %edx, (%rax)
        movq    -32(%rbp), %rax
        movl    -4(%rbp), %edx
        movl    %edx, (%rax)
        nop
        popq    %rbp
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE6:
        .size   swap2, .-swap2
        .section        .rodata
        .align 4
.LC0:
        .long   1069547520
        .align 4
.LC1:
        .long   1056964608
        .align 4
.LC2:
        .long   1092721050
        .align 4
.LC3:
        .long   1075838976
        .ident  "GCC: (Ubuntu 5.4.0-6ubuntu1~16.04.12) 5.4.0 20160609"
        .section        .note.GNU-stack,"",@progbits
```

20191538 PL_HW7

1. activation record 구조



어셈블리 코드를 보면 global variable은 data 영역에 저장되어 있다. parameter와 local variable은 위와같이 stack 영역에 저장된다. main 함수는 mov $0, %eax 에서 0을 넣어 있듯이 eax에 0이 저장되고 return한다.
add 함수는 movl %edi, -4(%rbp) 에 sum이 저장되고 movl -4(%rbp), %eax 를 통해 return 값으로 사용된다.

2. subprogram 상에서 parameter, local변수, global 변수 참조 방법 분석

add 함수를 살펴보면 movl %edi, -20(%rbp) 에서 parameter 값을 저장한다. 즉 activation record의 저장 위치로 local 변수도 activation record에 저장되어 있다. global 변수는 movl a(%rip), %edx 를 보면 a의 값을 edx를 통해 참조한다.

3. 다양한 데이터 타입에 대한 parameter passing

integer : movl %eax, %edi edi에 저장해두었다가 복사해온다.
float : movss -68(%rbp), %xmm0 실수값을 xmm0 레지스터로 조정한다. 이런 방법이 passing 이 이루어진다.
Array : leaq -16(%rbp), %rax leaq를 통해 배열의 시작 주소를 넘겨준다.
Structure : movq %rdx, %rdi, movq %rax, %rsi 구조체 매개변수는 %rdi와 %rsi 레지스터를 통해 전달된다.

4. Return value 전달방식 분석

add 함수의 movl -4(%rbp), %eax 보면 integer은 %eax에 값을 저장해서 return한다.
float는 %xmm0에 저장해서 return한다.
void는 nop no operation으로 return 안한다.
swap1을 보면 movl %edx, %esi, movl %eax, %edi로, esi, edi로 받아온다.
swap2는 leaq -56(%rbp), %rdx, leaq -60(%rbp) %rax로 leaq를 통해 %rdx, %rsi 주소를 받아온다.
이처럼 swap2는 주소를 받아옴으로 값이 바뀐다.