```c
#include <stdio.h>

int arr_len = 5;

typedef struct _myStruct{
    float f;
    char c;
    int i;
}MyStruct;


int AddArr(int *arr){
    int sum = 0;
    for(int i = 0 ; i < arr_len; i++){
        sum += arr[i];
    }
    return sum;
}

float AddStruct(MyStruct myStruct){
    MyStruct m = myStruct;
    float sum = myStruct.f + myStruct.c + myStruct.i;
    return sum;
}

float Add(int i, float f){
    int ii = i;
    float ff = f;
    ff = i + f;
    return ff;
}
```

```c
int main(){
    int myint = 1;
    float myfloat = 2.0f;
    int myArr[] = {0,1,2,3,4};
    MyStruct myStruct;

    int arrSum = AddArr(myArr);
    float structSum = AddStruct(myStruct);
    float sum1 = Add(arrSum, structSum);
    float sum2 = Add(myint, myfloat);
    float finalSum = Add((int)sum2, sum1);
    printf("%lf", finalSum);
    return 0;
}
```

```
    .file   "test.c"
    .globl  arr_len
    .data
    .align 4
    .type   arr_len, @object
    .size   arr_len, 4
arr_len:
    .long   5
    .text
    .globl  AddArr
    .type   AddArr, @function
AddArr:
.LFB0:
    .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)
    movl    $0, -8(%rbp)
    movl    $0, -4(%rbp)
    jmp .L2
.L3:
    movl    -4(%rbp), %eax
    cltq
    leaq    0(,%rax,4), %rdx
    movq    -24(%rbp), %rax
    addq    %rdx, %rax
    movl    (%rax), %eax
    addl    %eax, -8(%rbp)
    addl    $1, -4(%rbp)
```

```
.L2:
    movl    arr_len(%rip), %eax
    cmpl    %eax, -4(%rbp)
    jl  .L3
    movl    -8(%rbp), %eax
    popq    %rbp
    .cfi_def_cfa 7, 8
    ret
    .cfi_endproc
.LFE0:
    .size   AddArr, .-AddArr
    .globl  AddStruct
    .type   AddStruct, @function
AddStruct:
.LFB1:
    .cfi_startproc
    pushq   %rbp
    .cfi_def_cfa_offset 16
    .cfi_offset 6, -16
    movq    %rsp, %rbp
    .cfi_def_cfa_register 6
    movq    %rdi, %rdx
    movl    %esi, %eax
    movq    %rdx, -48(%rbp)
    movl    %eax, -40(%rbp)
    movq    -48(%rbp), %rax
    movq    %rax, -16(%rbp)
    movl    -40(%rbp), %eax
    movl    %eax, -8(%rbp)
    movss   -48(%rbp), %xmm1
    movzbl  -44(%rbp), %eax
    movsbl  %al, %eax
    pxor    %xmm0, %xmm0
    cvtsi2ss    %eax, %xmm0
    addss   %xmm0, %xmm1
    movl    -40(%rbp), %eax
    pxor    %xmm0, %xmm0
    cvtsi2ss    %eax, %xmm0
    addss   %xmm1, %xmm0
    movss   %xmm0, -20(%rbp)
```

```
    movss   -20(%rbp), %xmm0
    popq    %rbp
    .cfi_def_cfa 7, 8
    ret
    .cfi_endproc
.LFE1:
    .size   AddStruct, .-AddStruct
    .globl  Add
    .type   Add, @function
Add:
.LFB2:
    .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)
    movss   %xmm0, -24(%rbp)
    movl    -20(%rbp), %eax
    movl    %eax, -8(%rbp)
    movss   -24(%rbp), %xmm0
    movss   %xmm0, -4(%rbp)
    pxor    %xmm0, %xmm0
    cvtsi2ss    -20(%rbp), %xmm0
    addss   -24(%rbp), %xmm0
    movss   %xmm0, -4(%rbp)
    movss   -4(%rbp), %xmm0
    popq    %rbp
    .cfi_def_cfa 7, 8
    ret
    .cfi_endproc
.LFE2:
    .size   Add, .-Add
    .section    .rodata
```

```
.LC1:
    .string "%lf"
    .text
    .globl  main
    .type   main, @function
main:
.LFB3:
    .cfi_startproc
    pushq   %rbp
    .cfi_def_cfa_offset 16
    .cfi_offset 6, -16
    movq    %rsp, %rbp
    .cfi_def_cfa_register 6
    subq    $96, %rsp
    movq    %fs:40, %rax
    movq    %rax, -8(%rbp)
    xorl    %eax, %eax
    movl    $1, -76(%rbp)
    movss   .LC0(%rip), %xmm0
    movss   %xmm0, -72(%rbp)
    movl    $0, -32(%rbp)
    movl    $1, -28(%rbp)
    movl    $2, -24(%rbp)
    movl    $3, -20(%rbp)
    movl    $4, -16(%rbp)
    leaq    -32(%rbp), %rax
    movq    %rax, %rdi
    call    AddArr
    movl    %eax, -68(%rbp)
    movq    -48(%rbp), %rdx
    movl    -40(%rbp), %eax
    movq    %rdx, %rdi
    movl    %eax, %esi
    call    AddStruct
```

```
    movd    %xmm0, %eax
    movl    %eax, -64(%rbp)
    movl    -64(%rbp), %edx
    movl    -68(%rbp), %eax
    movl    %edx, -84(%rbp)
    movss   -84(%rbp), %xmm0
    movl    %eax, %edi
    call    Add
    movd    %xmm0, %eax
    movl    %eax, -60(%rbp)
    movl    -72(%rbp), %edx
    movl    -76(%rbp), %eax
    movl    %edx, -84(%rbp)
    movss   -84(%rbp), %xmm0
    movl    %eax, %edi
    call    Add
    movd    %xmm0, %eax
    movl    %eax, -56(%rbp)
    movss   -56(%rbp), %xmm0
    cvttss2si   %xmm0, %eax
    movl    -60(%rbp), %edx
    movl    %edx, -84(%rbp)
    movss   -84(%rbp), %xmm0
    movl    %eax, %edi
    call    Add
    movd    %xmm0, %eax
    movl    %eax, -52(%rbp)
    cvtss2sd    -52(%rbp), %xmm0
    movl    $.LC1, %edi
    movl    $1, %eax
    call    printf
    movl    $0, %eax
    movq    -8(%rbp), %rcx
    xorq    %fs:40, %rcx
    je  .L11
    call    __stack_chk_fail
```

```
.L11:
    leave
    .cfi_def_cfa 7, 8
    ret
    .cfi_endproc
.LFE3:
    .size   main, .-main
    .section    .rodata
    .align 4
.LC0:
    .long   1073741824
    .ident  "GCC: (Ubuntu 5.4.0-6ubuntu1~16.04.12) 5.4.0 20160609"
    .section    .note.GNU-stack,"",@progbits
```

2016120 유창호

1. Activation Record 구조

프로그램에서는 stack pointer를 감소시켜 해당 함수의 지역 변수, return 주소, return 값을 저장하고, 사용한 다음에는 stack pointer를 증가시켜 stack을 pop 하는 것을 알 수 있다.

예를 들어 Add 함수를 실행할 때 pushq %rbp 로 return address를 저장한다. 그 후 두 지역 변수를 movq, movl을 사용하여 stack에 넣는다. Add가 끝나고 다시 원래 위치로 반환될 때 popq %rbp을 사용해 rbp 값에 현재 rsp 값을 넣어주고 stack pointer를 증가시킨다.

2. Subprogram에서 parameter, local 변수, global 변수 참조.

Subprogram으로 parameter를 전달할 때는 stack을 사용한다. Add 함수에서 int, float 타입 매개 변수는 넘겨주는데, 이때 시스템 상으로 int는 4byte, float는 4byte로 생각해서 각각 rbp 값을 -4 감소시켜 공간을 확보한 다음 그곳에 값을 복사한다. 매개 변수를 복사하기 때문에 call by value 인 것을 알 수 있다.

local 변수는 레지스터로 저장된다. Add 함수에서 매개 변수를 지역 변수 ii, ff에 저장할 때 %edi, %xmm0 레지스터의 값을 복사해 온다.

global 변수는 data 영역에 저장되어 참조할 때는 변수 이름 그대로 사용한다. AddArr 함수에서 arr_len 변수를 사용할 때 movl arr_len(%rip), %eax 명령을 통해 global 변수를 참조한다.

3. Int, Float, Array, Struct 패싱.

Int, Float — Add 함수에서 본 것처럼 stack pointer를 증가시키고 (int float 4byte) 해당 위치에 값 복사.

Array — AddArr 함수에서 배열을 넘겨줄 때 인덱스 레지스터 %rsi를 사용한다.

Struct — AddStruct 함수에서 stack을 사용해 struct를 넘겨준 것을 알 수 있다. 각 멤버 값을 참조할 때는 멤버 타입의 크기와 word alignment에 맞춰서 stack pointer을 조정하고 값을 가져온다.

4. Return Value

반환 값은 return register 인 rax(eax)에 저장되어 return 된 곳에서 이 레지스터에 저장된 값을 확인할 수 있다.