Programming Language – HW3

20191559 강상원

main.c

```c
#include <stdio.h>
#include <stdbool.h>

int arr[8][9][10];

enum Months {
    January, February, March, April, May, June, July,
    August, September, October, November, December,
} months;

struct {
    int i;
    char c1;
    float f;
    char c2;
} st;

union UNION {
    double d;
    char cc;
    bool b;
} un;

int *pointer_int;
char* pointer_char;
int integer;

int main() {
    pointer_int = &integer;
    pointer_char = &st.c2;
    integer = arr[4][5][7];
    un.b = true;
    un.d = 3.14;

    return 0;
}
```

main.s

```
        .file   "main.c"
        .comm   arr,2880,32
        .comm   months,4,4
        .comm   st,16,16
        .comm   un,8,8
        .comm   pointer_int,8,8
        .comm   pointer_char,8,8
        .comm   integer,4,4
        .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
        movq    $integer, pointer_int(%rip)
        movq    $st+12, pointer_char(%rip)
        movl    arr+1668(%rip), %eax
        movl    %eax, integer(%rip)
        movb    $1, un(%rip)
        movsd   .LC0(%rip), %xmm0
        movsd   %xmm0, un(%rip)
        movl    $0, %eax
        popq    %rbp
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE0:
        .size   main, .-main
        .section        .rodata
        .align 8
.LC0:
        .long   1374389535
        .long   1074339512
        .ident  "GCC: (Ubuntu 5.4.0-6ubuntu1~16.04.12) 5.4.0 20160609"
        .section        .note.GNU-stack,"",@progbits
```

1. .file "main.c" : 이 파일이 "main.c"라는 이름의 C 소스 파일에서 생성되었음을 나타낸다.

2~8. .comm : 전역 변수를 선언하고 초기화한다. 변수명, 크기(byte), 정렬(byte) 순서로 나타낸다.

2 - .comm arr, 2880, 32 : 이름은 arr이고, int[8][9][10]이므로 4×8×9×10 = 2880 byte를 할당한다.

3 - .comm months, 4, 4 : 이름은 months이고, enum이므로 4 byte가 할당된다. 그리고 word alignment가 4 byte 단위로 일어난다.

4 - .comm st, 16, 16 : 이름이 st이고, int, char, float, char 이므로 4 + 4(1+skip) + 4 + 4(1+skip) = 16 byte이다. Word alignment가 16 byte 단위로 일어난다.

5 - .comm un, 8, 8 : 이름이 un이고, double이 union 中 가장 크므로 8 byte가 할당된다. Word alignment 4 byte 단위로 발생.

6 - .comm pointer_int, 8, 8 : 이름이 pointer_int이고, 포인터이기 때문에 8 byte 할당. Word alignment 8 byte 단위.

7 - .comm pointer_char, 8, 8 : 이름이 pointer_char, 마찬가지로 포인터여서 8 byte. Word alignment 8 byte.

8 - .comm integer, 4, 4 : 이름이 integer이고, int 크기 4 byte 할당. Word alignment 4 byte 단위.

9. .text : 코드 section을 시작한다. 이 section에는 실행 가능한 명령어들이 포함된다.

10. .globl main : 'main' 함수가 전역으로 사용 가능함을 나타낸다.

11. .type main, @function : 'main'이 함수임을 가타낸다.

12 main : 'main' 함수의 시작 지점을 나타낸다.

13. .LFB0 : LFB0 레이블

14. .cfi_startproc : 디버깅을 위해, call-frame_information을 저장하는 기능을 시작한다. 오류 occur 시, stack unwinds 오류 발생 위치를 찾을 수 있다.

15 pushq %rbp : rbp 레지스터 값을 stack에 push.

16. .cfi_def_cfa_offset 16 : CFA (Call Frame Address) 오프셋을 16으로 정의한다.

17. .cfi_offset 6, -16 : CFI (Call Frame Information) 데이터를 업데이트하여 레지스터 %rbp (6번 레지스터)의 오프셋을 -16으로 지정한다.

18 movq %rsp, %rbp : 스택 포인터 값을 베이스 포인터에 복사한다.

19. .cfi_def_cfa_register 6 : CFA의 기준 레지스터를 %rbp (6번 레지스터)로 변경한다.

20 movq $integer, pointer_int(%rip) : 'integer' 변수의 주소를 'pointer_int'에 저장한다.

21 movq $st+12, pointer_char(%rip) : 구체적 'st'의 'c2' 멤버의 주소를 'pointer_char'에 저장한다.

22 movl arr+1668(%rip), %eax : 배열 'arr'의 원소 arr[4][5][7] 값을 %eax 레지스터에 저장한다.

$(4\times 9\times 10 + 5\times 10 + 7)\times 4 = 1668$

23 movl %eax, integer(%rip) : %eax 레지스터의 값을 'integer' 변수에 저장한다.

24 movb $1, un(%rip) : 'un'의 'b' 멤버에 'true'(1) 값을 저장한다.

25 movsd .LC0(%rip), %xmm0 : 상수 '3.14'를 %xmm0 레지스터에 저장한다.

26 movsd %xmm0, un(%rip) : %xmm0 레지스터의 값을 'un'의 'd' 멤버에 저장한다.

27 movl $0, %eax : %eax 레지스터에 0을 저장한다. 이는 'main' 함수의 반환값을 나타낸다.

28 popq %rbp : 스택에서 베이스 포인터 값을 복원한다.

29 .cfi_def_cfa 7, 8 : CFA (Call Frame Address)를 정의한다. %rsp(7번 레지스터)를 기준으로 오프셋 8을 사용한다.

30 ret : 함수를 종료하고, 이전 호출자로 돌아간다.

31 .cfi_endproc : 함수 끝을 나타내는 directive.

32 .LFE0 : Local Label 정의.

33 .size main, .-main : 'main' 함수의 크기를 계산하여 지정한다.

34 .section .rodata : 읽기 전용 데이터 section 시작.

35 .align 8 : 다음 데이터를 8byte 경계에 맞춘다.

36 .LC0 : Local Label 정의.

37 .long 1374389535 : 8byte 실수 상수 '3.14'의 하위 32비트를 저장한다.

38 .long 1074339512 : 8byte 실수 상수 '3.14'의 상위 32비트를 저장한다.

39 .ident "GCC (Ubuntu ....)" ; 컴파일러 버전 정보

40 .section .note.GNU-stack, "", @progbits : 빈 스택 section을 생성해 명령어와 데이터를 구분하고, 실행 스택이 안전하게 생성되도록 한다.