리버스 엔지니어링

어셈블리 언어

1학년 전유경

목차

1. 어셈블리어

어셈블리어

0과 1을 대신한 사람이 이해하기 쉬운 저급 언어

기계어 : 어셈블리어 = 1 : 1

1. 어셈블리어

Windows - Intel

Linux - AT&T

공통점

Opcode operand1, operand2

명령 코드

피연산자

1. 어셈블리어

차이점

Intel : ADD operand1, operand2

1. 어셈블리어

차이점

AT&T : ADD operand1, operand2

1. 어셈블리어

	Intel	AT&T
숫자	1, 2, 3, 4, 5	$1, $2, $3, $4, $5
레지스터	EAX, EBX, ECX	%EAX, %EBX, %ECX
메모리 주소	[EAX]	(EAX)
오프셋	[EAX+4]	4(EAX)

2. 어셈블리어 명령어

명령어	설명
PUSH	스택에 값을 넣어줌
POP	스택 맨 위의 값을 가져옴
MOV	값을 옮겨줌 (복사)
LEA	주소값을 옮겨줌
INC	레지스터의 값을 증가
DEC	레지스터의 값을 감소
NOP	아무것도 하지 않음

ADD	값을 덧셈
SUB	값을 뺄셈
CALL	프로시저를 호출함
RET	CALL한 지점으로 돌아감
CMP	오퍼랜드를 비교함

3. 레지스터

레지스터

프로세서가 필요한 데이터를 저장하는 메모리

범용 레지스터

RAX : 연산에 사용, 함수의 return값 저장
RBX : 메모리 주소 저장
RCX : 반복문의 카운팅 변수 역할
RDX : 연산에 사용, 함수 return값 저장 X

3. 레지스터

인덱스 레지스터

RSI : 복사할 데이터의 주소 저장

RDI : 복사된 Destination의 주소 저장

포인터 레지스터

EBP : 스택 프레임의 시작 지점 주소 저장

ESP : 스택 프레임의 끝 지점 주소 저장

3. 레지스터

AX	EAX	RAX
BX	EBX	RBX
CX	ECX	RCX
DX	EDX	RDX
⇩	⇩	⇩
16 bit	32 bit	64 bit

4. 실습

```c
#include <stdio.h>

int main() {
    int a, b;

    a = 1;
    b = 10;

    a = a + b;

    printf("%d", a);
}
```

```asm
.LC0:
        .string "%d"
main:
        push    rbp
        mov     rbp, rsp
        sub     rsp, 16
        mov     DWORD PTR [rbp-4], 1
        mov     DWORD PTR [rbp-8], 10
        mov     eax, DWORD PTR [rbp-8]
        add     DWORD PTR [rbp-4], eax
        mov     eax, DWORD PTR [rbp-4]
        mov     esi, eax
        mov     edi, OFFSET FLAT:.LC0
        mov     eax, 0
        call    printf
        mov     eax, 0
        leave
        ret
```

4. 실습

```c
#include <stdio.h>

int main() {
    int a, b, c;
    a = 1;
    b = 5;
    scanf("%d", &c);

    if (a<c && c<b){
        printf("success");
    }
    else {
        printf("fail");
    }

}
```

4. 실습

```c
#include <stdio.h>

int main() {
    int a, b, c;
    a = 1;
    b = 5;
    scanf("%d", &c);
```

```
.LC0:
        .string "%d"
.LC1:
        .string "success"
.LC2:
        .string "fail"
main:
        push    rbp
        mov     rbp, rsp
        sub     rsp, 16
        mov     DWORD PTR [rbp-4], 1
        mov     DWORD PTR [rbp-8], 5
        lea     rax, [rbp-12]
        mov     rsi, rax
        mov     edi, OFFSET FLAT:.LC0
        mov     eax, 0
        call    __isoc99_scanf
```

4. 실습

```
 9      if (a<c && c<b){
10          printf("success");
11      }
12      else {
13          printf("fail");
14      }
15
16  }
```

```
18          mov     eax, DWORD PTR [rbp-12]
19          cmp     DWORD PTR [rbp-4], eax
20          jge     .L2
21          mov     eax, DWORD PTR [rbp-12]
22          cmp     DWORD PTR [rbp-8], eax
23          jle     .L2
24          mov     edi, OFFSET FLAT:.LC1
25          mov     eax, 0
26          call    printf
27          jmp     .L3
28  .L2:
29          mov     edi, OFFSET FLAT:.LC2
30          mov     eax, 0
31          call    printf
32  .L3:
33          mov     eax, 0
34          leave
35          ret
```

감사합니다

4. 첨언

- 스택 프레임 공부
- 레지스터의 변화 과정 파악
- 포인트 레지스터 공부
- Jmp 종류 명령어 공부