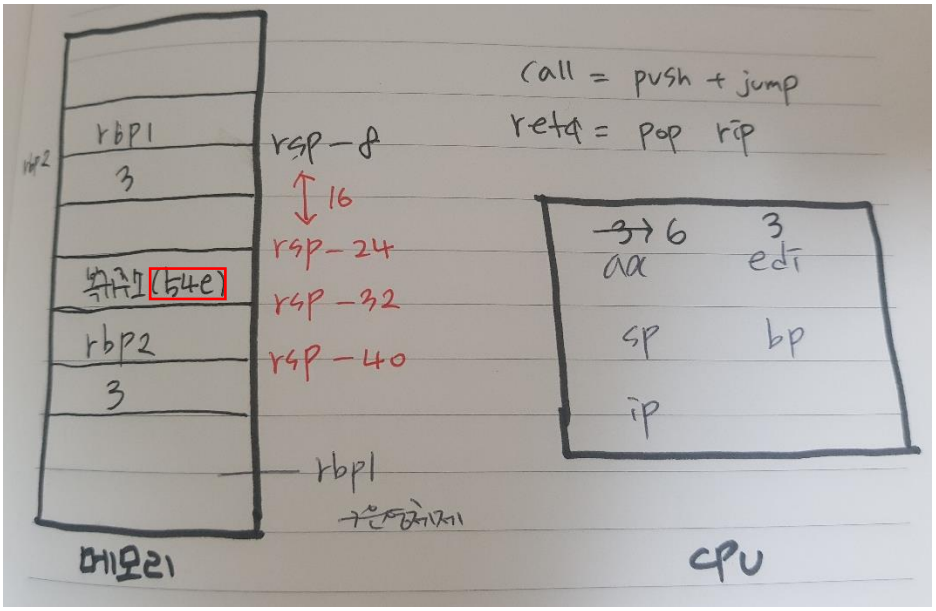
IOT 환경의 임베디드 개발자 양성과정

2일차 과제 문한나

1. 기계어 분석

```
(gdb) disas
Dump of assembler code for function main:
=> 0x0000000000400535 <+0>:     push   %rbp
   0x0000000000400536 <+1>:     mov    %rsp,%rbp
   0x0000000000400539 <+4>:     sub    $0x10,%rsp
   0x000000000040053d <+8>:     movl   $0x3,-0x8(%rbp)
   0x0000000000400544 <+15>:    mov    -0x8(%rbp),%eax
   0x0000000000400547 <+18>:    mov    %eax,%edi
   0x0000000000400549 <+20>:    callq  0x400526 <myfunc>
   0x000000000040054e <+25>:    mov    %eax,-0x4(%rbp)
   0x0000000000400551 <+28>:    mov    -0x4(%rbp),%eax
   0x0000000000400554 <+31>:    mov    %eax,%esi
   0x0000000000400556 <+33>:    mov    $0x4005f4,%edi
   0x000000000040055b <+38>:    mov    $0x0,%eax
   0x0000000000400560 <+43>:    callq  0x400400 <printf@plt>
   0x0000000000400565 <+48>:    mov    $0x0,%eax
   0x000000000040056a <+53>:    leaveq
   0x000000000040056b <+54>:    retq
End of assembler dump.
(gdb)
```

```
(gdb) disas myfunc
Dump of assembler code for function myfunc:
   0x0000000000400526 <+0>:     push   %rbp
   0x0000000000400527 <+1>:     mov    %rsp,%rbp
   0x000000000040052a <+4>:     mov    %edi,-0x4(%rbp)
   0x000000000040052d <+7>:     mov    -0x4(%rbp),%eax
   0x0000000000400530 <+10>:    add    $0x3,%eax
   0x0000000000400533 <+13>:    pop    %rbp
   0x0000000000400534 <+14>:    retq
End of assembler dump.
(gdb)
```



2. 포인터 크기 내용 정리

- 8 비트 시스템의 경우 1 byte

  16 비트는 2 byte

  32 비트는 4 byte

  64 비트는 8 byte

✓ 포인터는 메모리에 주소를 저장하는 공간으로써 비트가 표현할 수 있는 최대값을 저장할 수 있어야 한다. 그러므로 사용하는 시스템이 32비트인 경우는 최대치인 4바이트로, 64비트인 경우는 8바이트로 처리된다.

edit fork download copy

```c
#include <stdio.h>

int main(void) {
        printf("sizeof(int *) = %lu\n", sizeof(int *));
        printf("sizeof(double *) = %lu\n", sizeof(double *));
        printf("sizeof(float *) = %lu\n", sizeof(float *));
        return 0;
}
```

Success #stdin #stdout 0s 4560KB comments (0)

stdin copy

```
Standard input is empty
```

stdout copy

```
sizeof(int *) = 8
sizeof(double *) = 8
sizeof(float *) = 8
```

3. 2진수, 16진수 변환 정리

- 2진수란? 0과 1로 각 자릿수를 표시하는 진수
- 16진수란? 0~9, A~F까지 총 16개의 숫자나 문자를 사용하여 표시하는 진수

✓ 컴퓨터는 2진수만을 사용하지만 인간이 상대적으로 쉽게 볼 수 있도록 16진수가 사용됨

ex) 144를 2진수와 16진수로 변환해보자

1. 2진수 변환

128	64	32	16	8	4	2	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

144 = 128 + 16 이므로

해당하는 자릿수에 1을 배치하면 **1001 0000**

2. 16진수 변환

16진수는 한 자리에 16개가 온다. 따라서 4개씩 끊자!(진수가 4자리)

8421	8421
1001	0000
9	0

=> **0x90**

ex) 431를 2진수와 16진수로 변환해보자

1. 2진수 변환

256	128	64	32	16	8	4	2	1
2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

431 = 256 + 128 + 32 + 8 + 4 + 2 + 1 이므로

해당하는 자릿수에 1을 배치하면 **1 1010 1111**

2. 16진수 변환

8421	8421	8421	
0001	1010	1111	
1	A	F	=> **0x1AF**