1배운 내용 복습 (goto , 파이프라인 for문)

* for문

for문의 탄생 배경 ?

while(초기화 조건식 증감식)문을 간결화

초기화 int i=0;

조건부 while(i <10) ——> for(i=0 ;i<10; result ++) result++뒤에 ;안씀

result ++; 증감식

```
 1 #include<stdio.h>
 2 int main(void)
 3 {
 4         int i , result ;
 5
 6         for(i=0, result='A';i<10; i++, result++)
 7         {
 8         printf("%c\n",result);
 9         }
10         return 0;
11 }
```

```
a
b
c
d
e
f
g
```

h
i
j

*goto ,파이프라인

=> break 활용시

```
 1 #include<stdio.h>
 2
 3 int main(void)
 4 {
 5         int i, j, k , flag =0 ;
 6
 7         for(i = 0; i < 5; i++)
 8         {
 9                 for(j=0; j<5; j++)
10                 {
11                         for(k =0 ; k <5 ; k++)
12                         {
13                                 if((i ==2) &&(j ==2)&&(k ==2))
14                                 {
15                                         printf("Error!!!\n");
16                                         flag =1;
17                                 }
18                                 else
19                                 {
20                                         printf("data\n");
21
22                                 }
23                         if(flag)
```

```
24                                  break;
25                                  }
26                          if(flag)
27                          break;
28
29                          }
30
31              if(flag)
32              break;
33              }
34
35              return 0;
36 }
```

25+25+5+5

마지막 Error로 끝남

goto 활용

```
#include<stdio.h>
  2
  3 int main(void)
  4 {
  5              int i, j, k   ;
  6
  7              for(i = 0; i < 5; i++)
  8              {
  9                          for(j=0; j<5; j++)
```

```
10                  {
11                          for(k =0 ; k <5 ; k++)
12                          {
13                                  if((i ==2) &&(j ==2)&&(k ==2))
14                                  {
15                                          printf("Error!!!\n");
16                                          goto err_handler;
17                                  }
18                                  else
19                                  {
20                                          printf("data\n");
21
22                                  }
23

24                          }
25
26                  }
27
28          }
29
30          return 0;
31
32          err_handler:
33          printf("Goto zzang!\n");
34
35          return -1;
36 }
37
```

data 쭉

Error

Goto zzang!

*goto의 이점과 cpu 파이프라인

앞서서 만든 goto예제는

if 와 break를 조합한 버전과

goto로 처리하는 버전을 가지고 있따.

if문은 기본적으로 mov cmp jmp 로 구성된다
goto 는 jmp하나로 끝이다

for문이 여러개 생기면 if, break 조합의 경우
for 문의 갯수만큼 mov cmp jmp를 해야 한다.
문제는 바로 jmp 명령어다

call이나 jmp를 cpu instruction(명령어) 레벨에서 분기 명렁어라고 하고 이들은
cpu 파이프라인에 매우 치명적인 손실을 가져다준다.

기본적으로 아주 단순한
cpu의 파이프라인을 설명하자면
아래와 같은 3단계로 구성된다

1.fetch - 실행해야할 명령어를 물어옴
2.decode - 어떤 명령어인지 해석함
3. execute - 실제 명령어를 실행시킴

파이프라인이 짧은 것부터 긴 것이
5 단계 ~ 수십 단계로 구성된다.
(ARM , intel 등등 다양한 프로세서들 모두 마찬가지)

그런데 왜 jmp 나 call 등의 분기명령어가 문제가 될까?
기본적으로 분기 명령어는 파이프라인을 때려부순다.
이 뜻은 가장 단순한 위의 cpu가 실행까지 3clock을 소요하는데
파이프라인이 깨지니 쓸때없이 또 다시 3clock을 버려야함을 의미한다.

만약 파이프라인의 단계가 수십 단계라면
분기가 여러번 발생하면
파이프라인 단계 x 분기 횟수만큼
CPU clock을 낭비하게 된다.

즉 성능면에서도 goto가 월등히 압도적이다.
(jmp 1번에 끝나니까)