펭귄 스터디 내용06

작성자 : 류성수
작성일자 : 24.07.01

url : https://www.youtube.com/watch?v=TdzCyRviW7w&list=PLz-ENLG_8TMdMJIwyqDIpcEOysvNoonf&index=8

● 파이프라인

1. 컴퓨터는 왜 모든 것을 내부 clk를 이용해 처리하는가?

1. 1. 첫 번째 이유

컴퓨터에게는 시간의 개념이 없다. 그렇기에, 최소의 단위로 내부 clk를 사용한다.
이를 통해, CPU는 clk에 몇초가 걸렸는지에 구애받지 않고, 단지 clk의 몇사이클이 지났는지 만을 확인한다.

1. 2. 두 번째 이유

컴퓨터의 모든 HW 동기화를 맞춰야 한다. 예를 들어, 간단히 a = a + 3; 이라는 작업에도, 기계어 단에서 여러가지의 명령이 실행된다.
어셈블리 단에서는 총 3개의 명령으로 실행된다.

```
mov eax, a   ; a 값을 eax 레지스터로 로드
add eax, 3   ; eax에 3을 더함
mov a, eax   ; eax 값을 다시 a에 저장
```

이를 기계어 단에서 살펴보면 여러 가지의 명령으로 세분화 된다.

이름	설명
Fetch	mov eax, a 명령어를 메모리에서 가져옵니다.
Decode	mov eax, a 명령어를 해독하여, a의 값을 eax 레지스터로 이동하는 동작으로 이해합니다.
Execute	mov eax, a 명령어를 실행하여, 변수 a의 값을 eax 레지스터에 로드합니다. 다음으로, add eax, 3 명령어를 가져오고 해독한 후, eax 레지스터에 3을 더하는 연산을 수행합니다.
Memory	mov a, eax 명령어를 가져오고 해독한 후, eax 레지스터의 값을 다시 변수 a에 저장하기 위해 메모리에 접근합니다.
Write back	eax 레지스터의 값을 변수 a에 기록하여 최종 결과를 메모리에 저장합니다.

이 모든 기계어 단에서의 동작들에 1clk가 소모된다.
여기서, 중요한건, 메모리에서 가져오고, 더하고, 명령어를 해독하는 HW들이 서로 다르다는 것이다. 이들과 협업하여 명령을 수행해야 하기 때문에, clk가 필수적인 것이다.

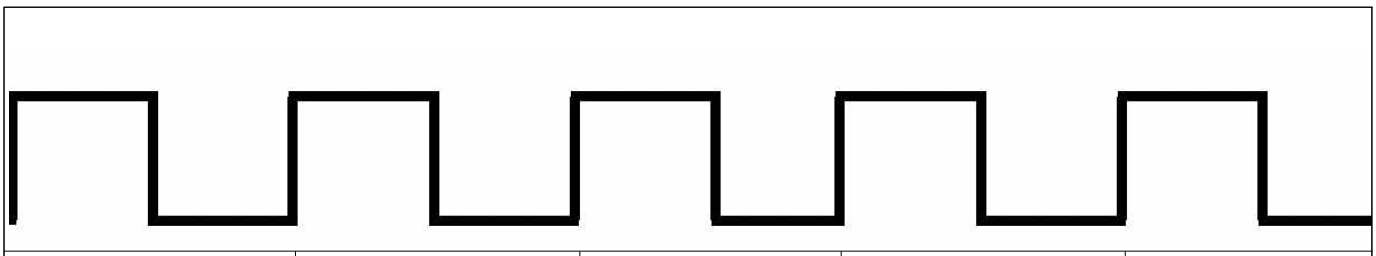
2. 파이프라인

아까처럼 a+=3; 의 명령을 처리할 때, CPU안의 독립적인 하드웨어들이 따로따로 움직인다는 점을 설명하였다.
독립적인 하드웨어들이기에, 소스의 코드 한 줄을 처리할 때, 순서만 지키면 되지, 모든 HW들이 이 순서가 끝날 때 까지 기다려도 되지 않아도 된다는 것이다.

그래서, clk의 관점에서 3가지의 명령이 일어나고, 해당 명령들을 처리하는 HW가 존재한다고 가정하면, (처리시 1clk 필요)

AHW : 메모리값을 레지스터로 가져오는 HW
BHW : 해당 레지스터의 값을 계산하는 HW
CHW : 연산이 끝난 레지스터값을 다시 지정된 메모리에 쓰는 HW

만약, a +=3; b += 4; c += 5;의 명령을 처리한다면, 동작순서를 엄격히 지키는 CPU구조라면, 총 9clk가 걸린다.
그러나, 파이프라인의 개념이 들어간다면,



A -> reg	reg + 3	reg -> A		
	B -> reg	reg + 4	reg -> B	
		C -> reg	reg + 5	reg -> C

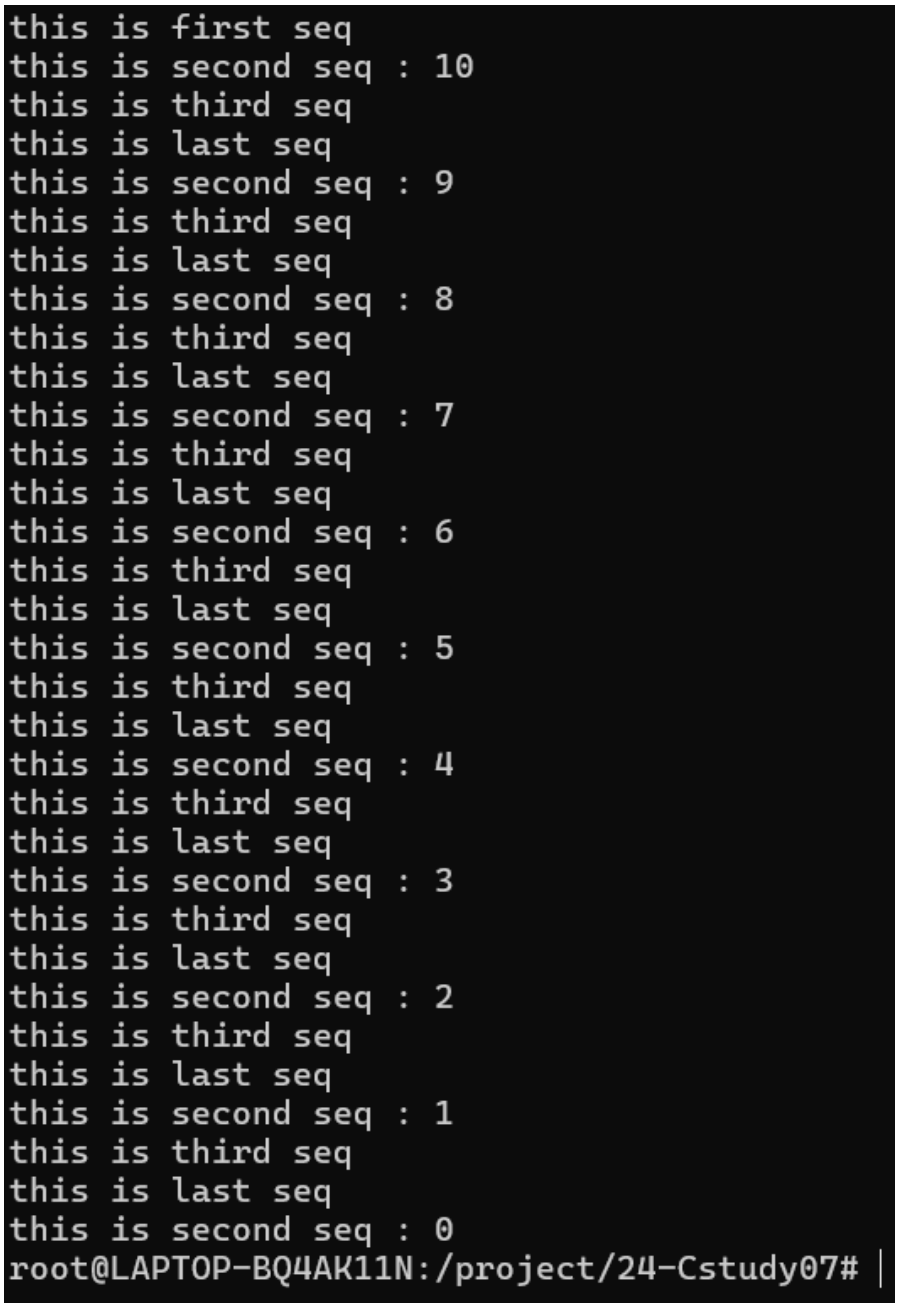
이렇게 총 5개의 clk로 끝낼 수 있다. 이는 연산에 필요한 리소스를 크게 줄여 매우 효율적이다.

● 반복문

항상 for문의 순서가 헷갈려서 오류를 빚는 일이 잦았는데, 이번기회에 for문의 순서를 직접 실습해보면서 파악하는 시간을 가졌다.

```c
#include <stdio.h>

int cnt = 10;

int first()
{
        printf("this is first seq\n");
        return 1;
}

int forth()
{
        printf("this is last seq\n");
        return 1;
}

int second()
{
        printf("this is second seq : %d\n",cnt);
        return cnt--;
}

int main()
{
        for(/*초기값*/first();/*조건식*/second();/*실행부분*/forth())
                printf("this is third seq\n");

        // do while 문 처럼 사용이 가능하다
        return 1;
}
```

이러한 코드를 실행했을 때, 다음과 같은 결과를 확인하였다.



그리고, for문의 초기값 부분에, 함수도 사용할 수 있기 때문에, for문을 do-while문으로 사용할 수도 있다는 아이디어를 얻었다.