

# 분기예측 (Branch prediction)

# 속도 비교(sorted VS not sorted)

```
const sum = () => {  
  const array = new Array(50000).fill(0);  
  for (let index in array) {  
    array[index] = Math.floor(Math.random() * 300);  
  }  
  let answer = 0;  
  
  array.sort();
```

```
  const startTime = Date.now();  
  for (let count = 0; count < 100000; count++) {  
    for (let num of array) {  
      if (num > 200) {  
        answer += num;  
      }  
    }  
  }  
}
```

```
  const endTime = Date.now();  
  console.log(answer);  
  console.log((endTime - startTime) / 1000);  
};
```

VS

```
const sum = () => {  
  const array = new Array(50000).fill(0);  
  for (let index in array) {  
    array[index] = Math.floor(Math.random() * 300);  
  }  
  let answer = 0;  
  
  // array.sort();
```

```
  const startTime = Date.now();  
  for (let count = 0; count < 100000; count++) {  
    for (let num of array) {  
      if (num > 200) {  
        answer += num;  
      }  
    }  
  }  
}
```

```
  const endTime = Date.now();  
  console.log(answer);  
  console.log((endTime - startTime) / 1000);  
};
```

# 속도 비교(sorted VS not sorted)

```
const sum = () => {  
  const array = new Array(50000).fill(0);  
  for (let index in array) {  
    array[index] = Math.floor(Math.random() * 300);  
  }  
  let answer = 0;  
  
  array.sort();
```

```
  const startTime = Date.now();  
  for (let count = 0; count < 100000; count++) {  
    for (let num of array) {  
      if (num > 200) {  
        answer += num;  
      }  
    }  
  }  
}
```

```
  const endTime = Date.now();  
  console.log(answer);  
  console.log((endTime - startTime) / 1000);  
};
```

```
> node sorted.js  
4 1 1 2 4 3 7 0 0 0 0  
9.503
```

VS

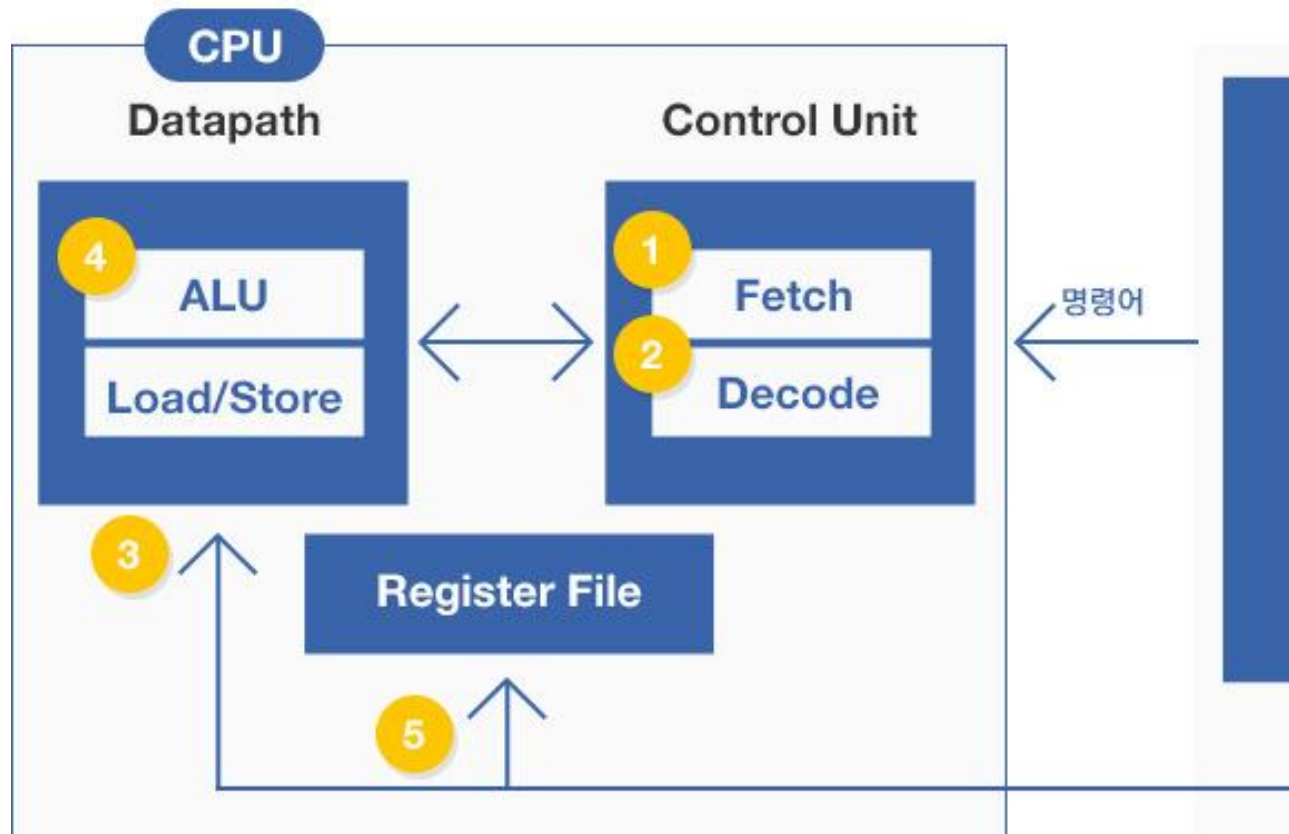
```
const sum = () => {  
  const array = new Array(50000).fill(0);  
  for (let index in array) {  
    array[index] = Math.floor(Math.random() * 300);  
  }  
  let answer = 0;  
  
  // array.sort();
```

```
  const startTime = Date.now();  
  for (let count = 0; count < 100000; count++) {  
    for (let num of array) {  
      if (num > 200) {  
        answer += num;  
      }  
    }  
  }  
}
```

```
  const endTime = Date.now();  
  console.log(answer);  
  console.log((endTime - startTime) / 1000);  
};
```

```
> node not_sorted.js  
4 1 1 0 1 1 5 0 0 0 0  
24.629
```

# CPU 구조



1. 명령어 가져오기(fetch)
2. 해석하기 (decode)
3. 연산에 필요한 데이터 메모리에서 가져오기(Memory Access)
4. 실제 연산(Execution)
5. 연산결과 저장(Write Back)

# No Pipeline



	1 clock	2 clock	3 clokc	4 clock	5 clock	6 clock
Fetch	명령어1					명령어2
Decode		명령어1				
Memory Access			명령어1			
Execution				명령어1		
Write Back					명령어1	

# Pipeline

	1 clock	2 clock	3 clock	4 clock	5 clock	6 clock
Fetch	명령어1	명령어2	명령어3	명령어4	명령어5	명령어6
Decode		명령어1	명령어2	명령어3	명령어4	명령어5
Memory Access			명령어1	명령어2	명령어3	명령어4
Execution				명령어1	명령어2	명령어3
Write Back					명령어1	명령어2

# 분기문 (ex IF, for, while)

```
if(num > 200) {  
  ① // 명령어 2  
    // 명령어 3  
    // 명령어 4  
    // 명령어 5  
    // 명령어 6  
}  
② // 명령어 7
```

	1	2	3	4	5	6
Fetch	IF	①/②?				
Decode		IF				
Memory Access						
Execution						
Write Back						

IF문의 연산 결과도 안나왔는데 어디로?? -> 확률은 50% 짚어보자

맞은 경우

	1 clock	2 clock	3 clock	4 clock	5 clock	6 clock
Fetch	IF	명령어 2	명령어 3	명령어 4	명령어 5	명령어6
Decode		IF	명령어 2	명령어 3	명령어 4	명령어5
Memory Access			IF	명령어 2	명령어 3	명령어4
Execution				IF	명령어 2	명령어3
Write Back					IF	명령어2



## 틀린 경우

	1 clock	2 clock	3 clock	4 clock	5 clock	6 clock
<b>Fetch</b>	IF	명령어 2	명령어 3	명령어 4	명령어 5	명령어7
<b>Decode</b>		IF	명령어 2	명령어 3	명령어 4	
<b>Memory Access</b>			IF	명령어 2	명령어 3	
<b>Execution</b>				IF	명령어 2	
<b>Write Back</b>					IF	

명령어 2,3,4,5는 불필요한 연산  
=> 시간이 지연됨!

# sorted

```
const sum = () => {  
  const array = new Array(50000).fill(0);  
  for (let index in array) {  
    array[index] = Math.floor(Math.random() * 300);  
  }  
  let answer = 0;  
  
  array.sort();  
  
  const startTime = Date.now();  
  for (let count = 0; count < 100000; count++) {  
    for (let num of array) {  
      if (num > 200) {  
        answer += num;  
      }  
    }  
  }  
  const endTime = Date.now();  
  console.log(answer);  
  console.log((endTime - startTime) / 1000);  
};
```

D	0	3	6	8	10	...	201	208	220	225
B	N	N	N	N	N		T	T	T	T



예측하기가 쉽다.

예측이 계속 적중하여서 시간 지연이 발생하지 않음

# Not sorted

```
const sum = () => {  
  const array = new Array(50000).fill(0);  
  for (let index in array) {  
    array[index] = Math.floor(Math.random() * 300);  
  }  
  let answer = 0;  
  
  // array.sort();  
  
  const startTime = Date.now();  
  for (let count = 0; count < 100000; count++) {  
    for (let num of array) {  
      if (num > 200) {  
        answer += num;  
      }  
    }  
  }  
  const endTime = Date.now();  
  console.log(answer);  
  console.log((endTime - startTime) / 1000);  
};
```

D	203	3	6	8	335	...	3 8	99	550	400
B	T	N	N	N	T		N	N	T	T




예측하기가 어렵다.

예측이 틀릴확률이 높음 => 틀린 만큼 시간 지연이 발생

# Pipeline Harzard

1. structural harzard
2. Data harzard
3. Control Harzard
  - Branch prediction



Q & A

