Computer Architecture & Real-Time Operating System

7. Processor Architecture (2/2)

(Microarchitecture)

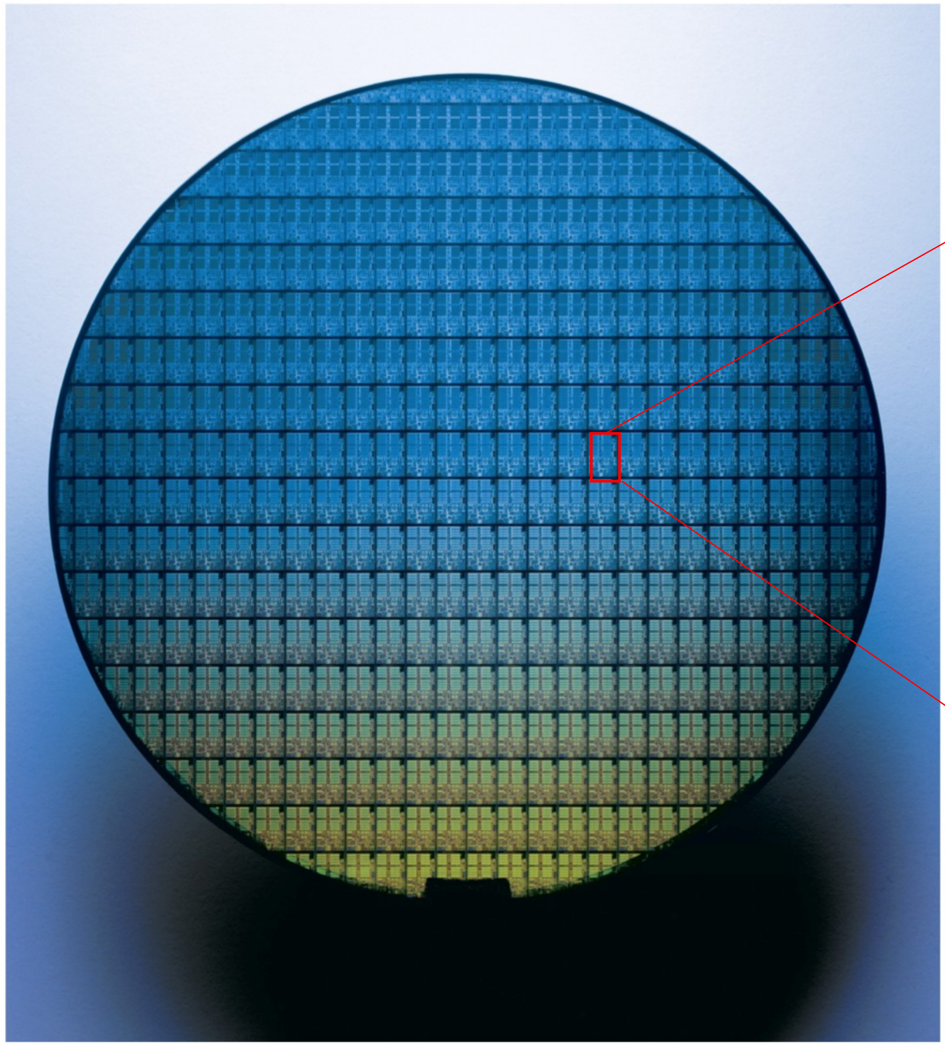
Prof. Jong-Chan Kim

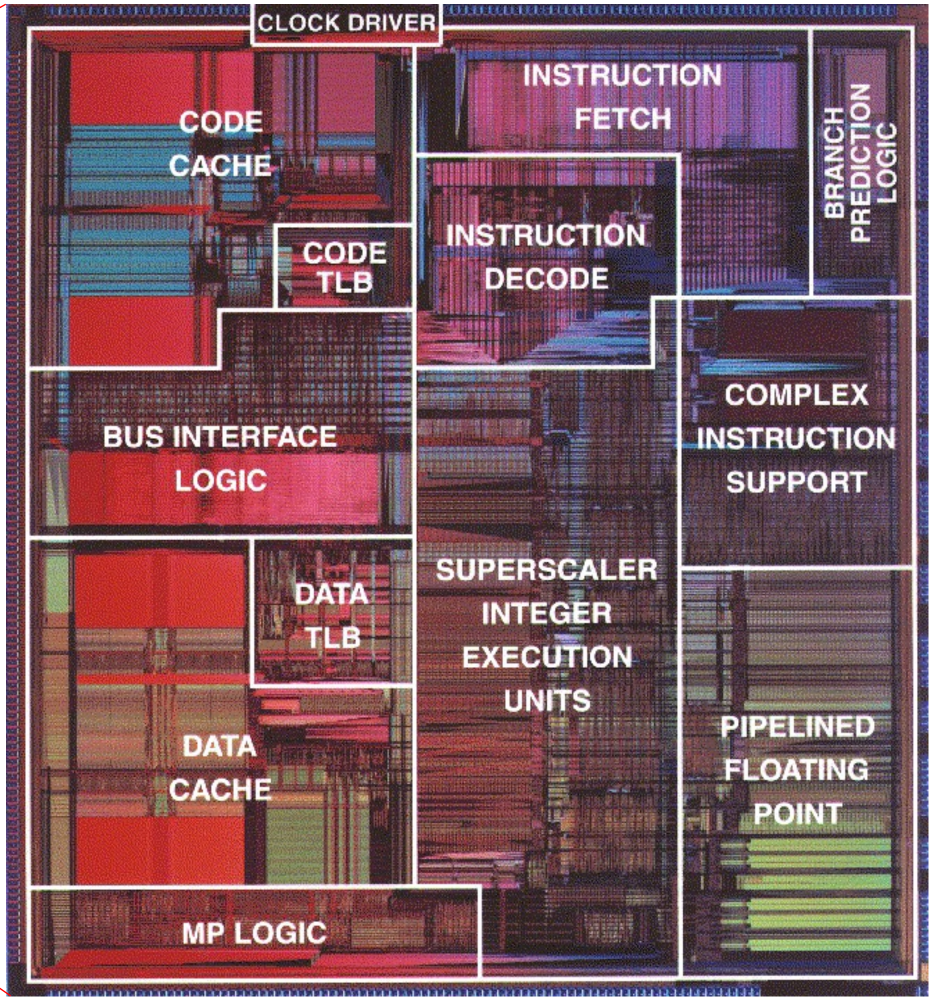
Dept. Automobile and IT Convergence

KMU 국민대학교
KOOKMIN UNIVERSITY

Microarchitecture

- What’s happening inside a chip?

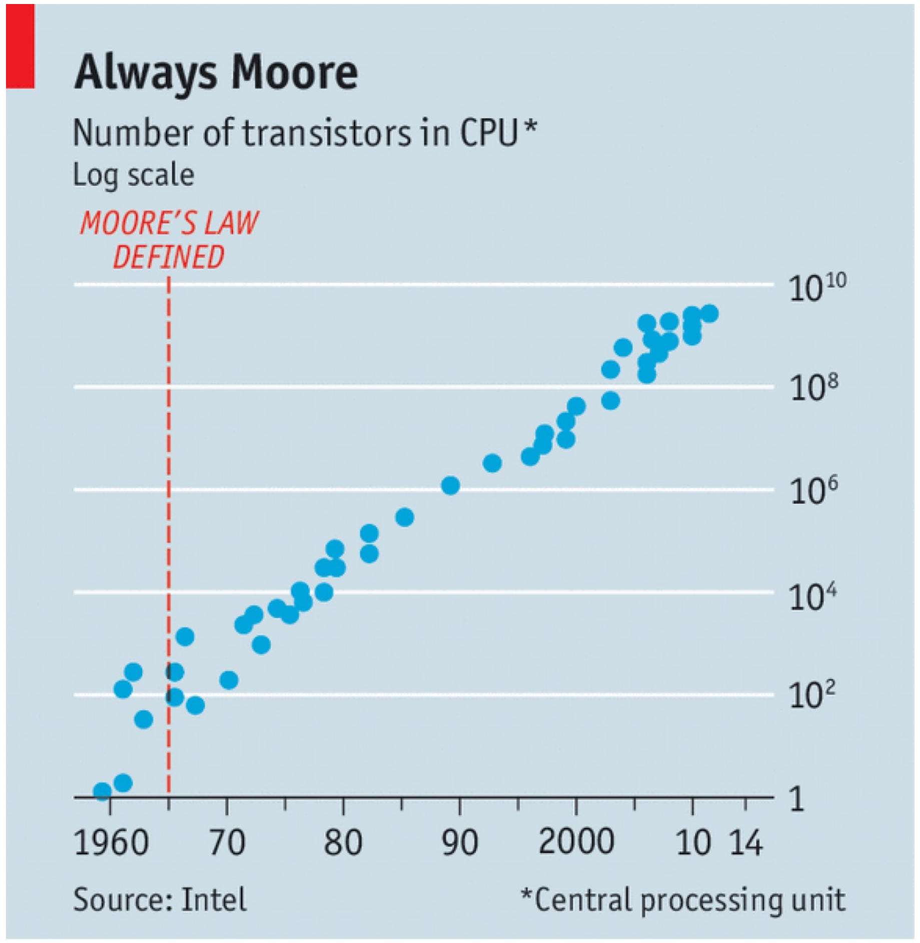




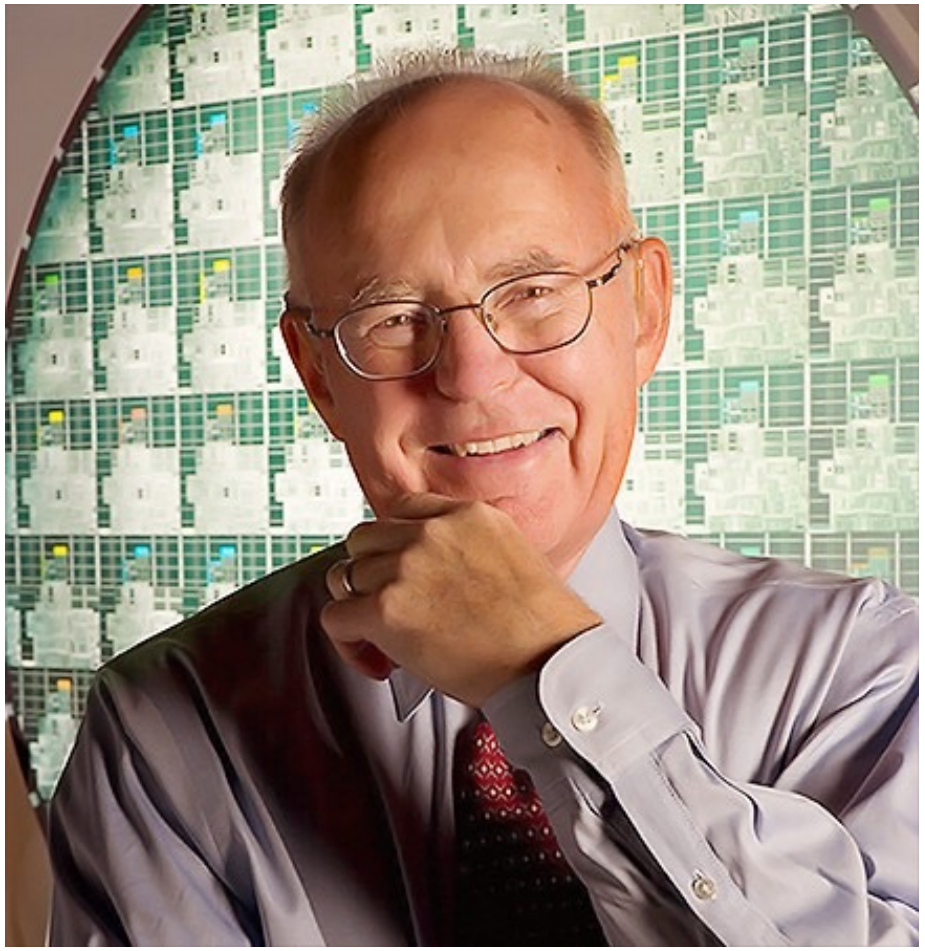
Die photo of Intel Pentium Processor

Moore's Law

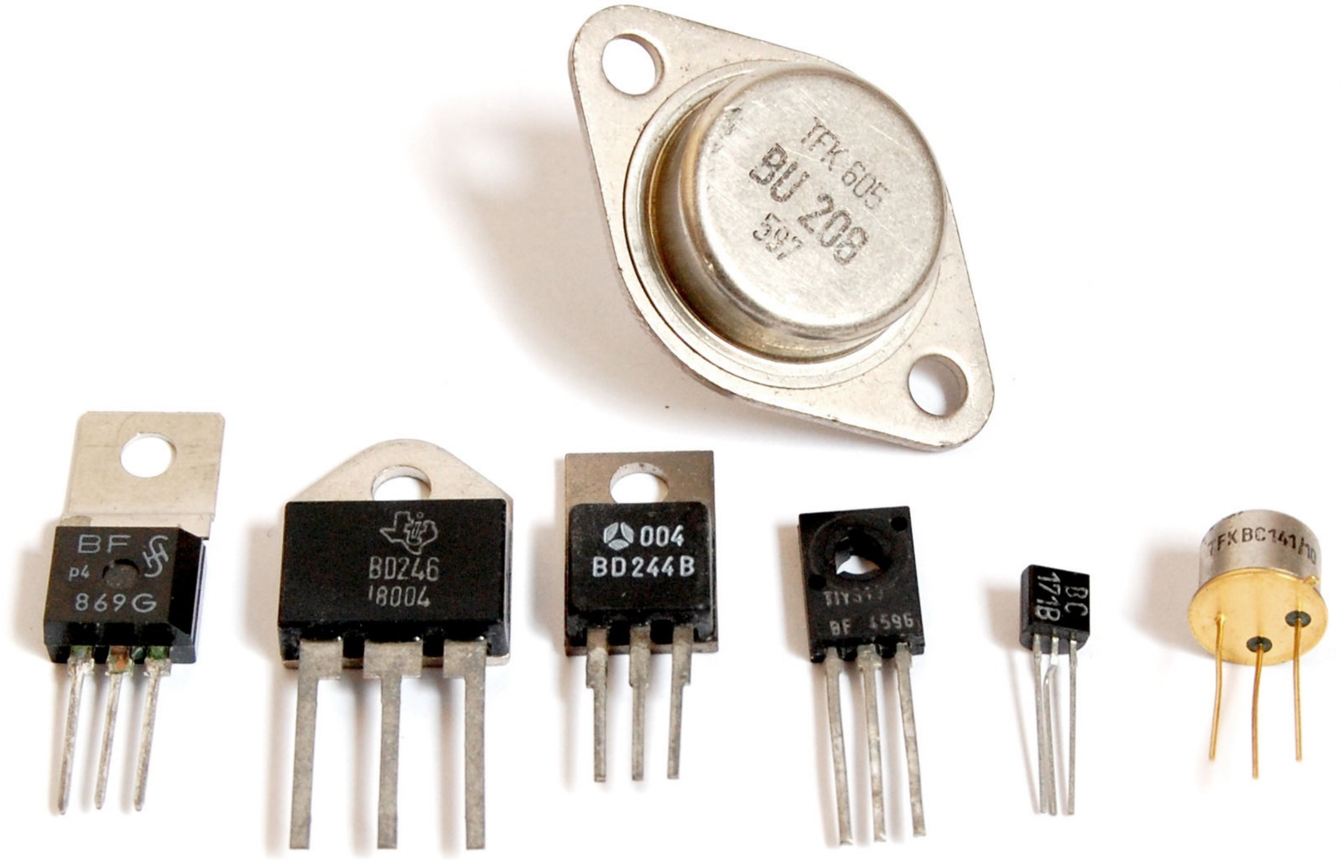
- Number of transistors in a CPU doubles every 24 months



Economist.com

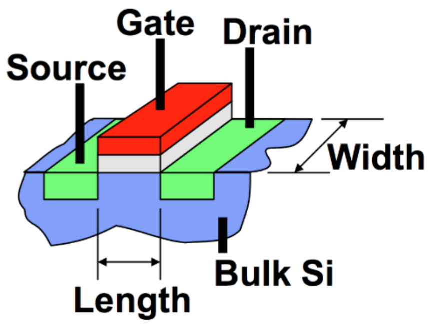

Gordon Moore (Co-founder of Intel)

Transistor

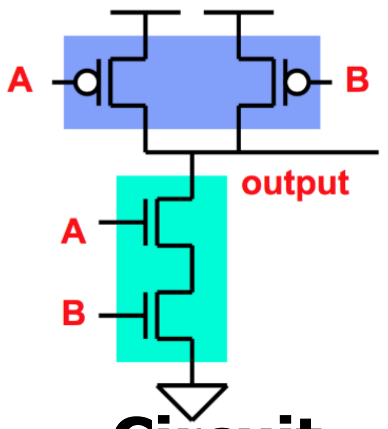


Source: https://ko.wikipedia.org/wiki/트랜지스터

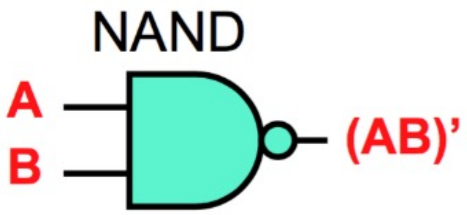
From Transistors to CPU



Transistor

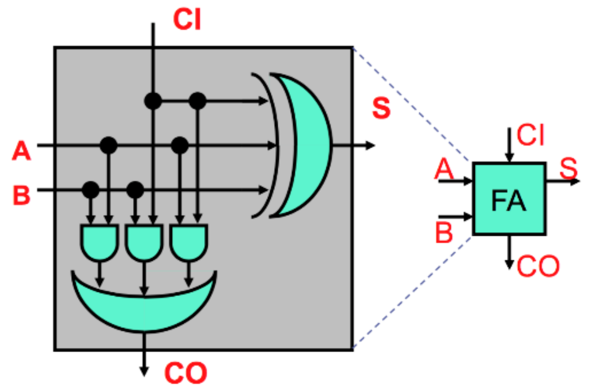


Circuit



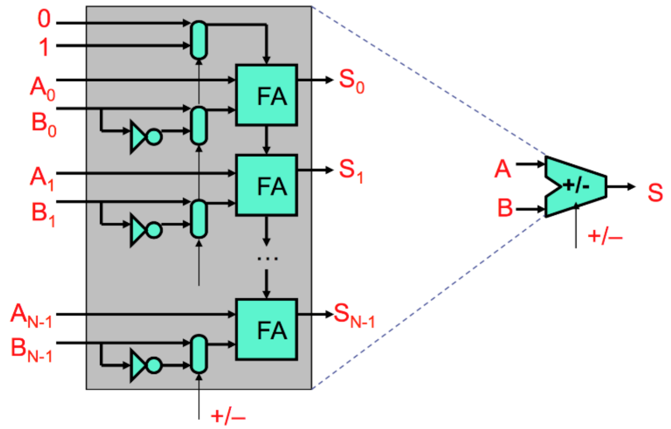
Logic Gate

Full Adder

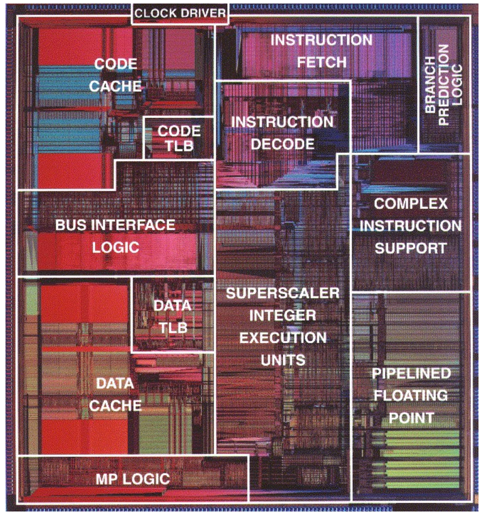


Module

N-bit Adder/Subtractor



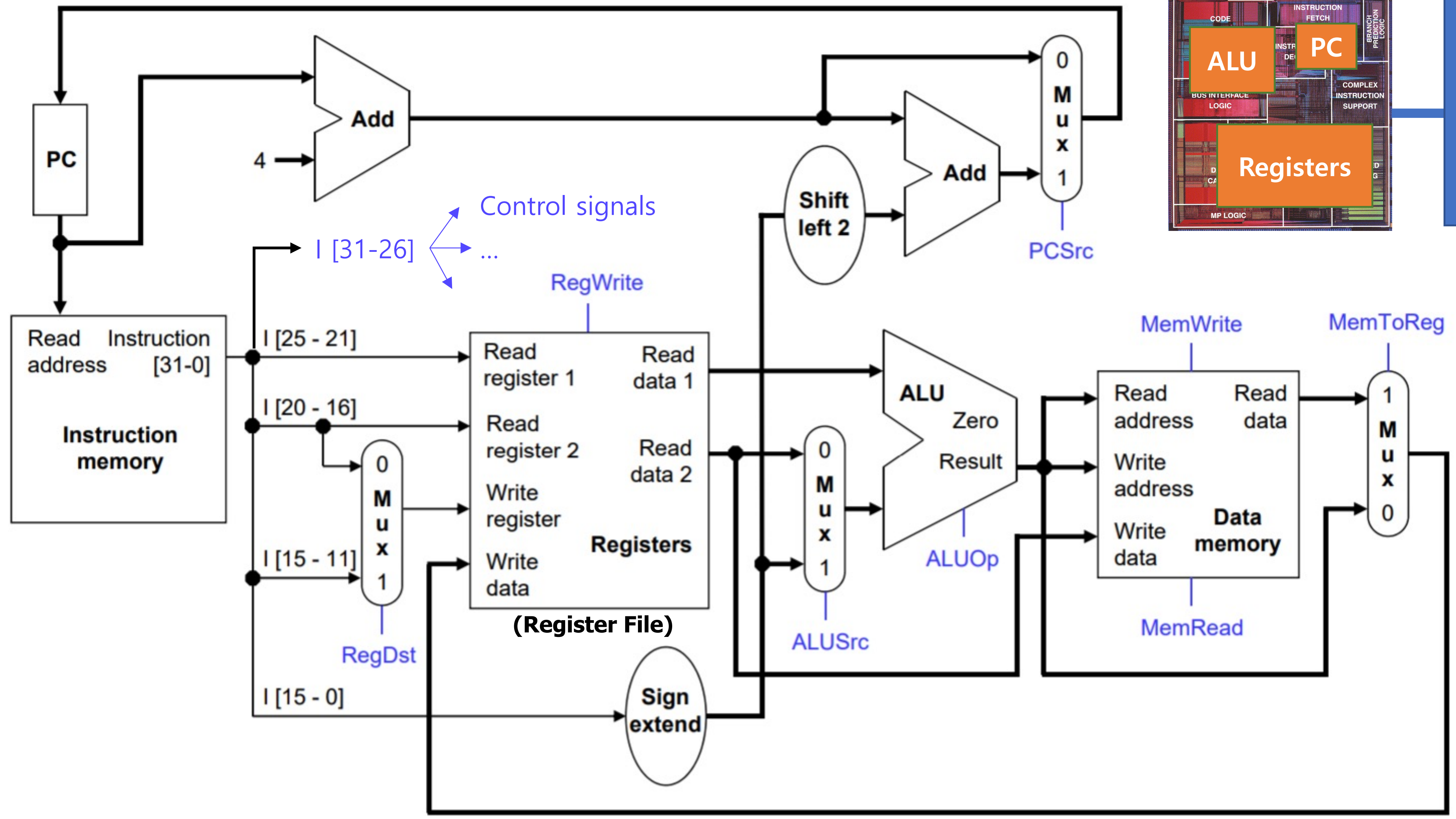
More Complex Module

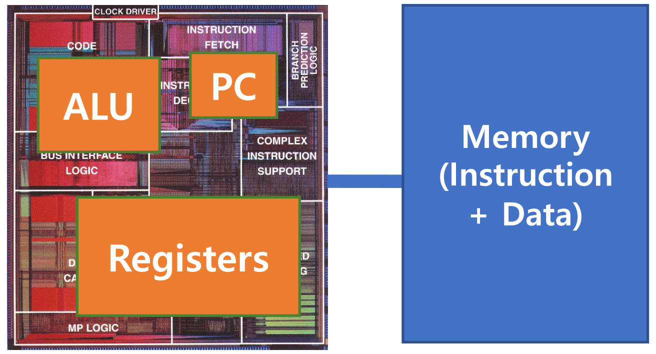


CPU

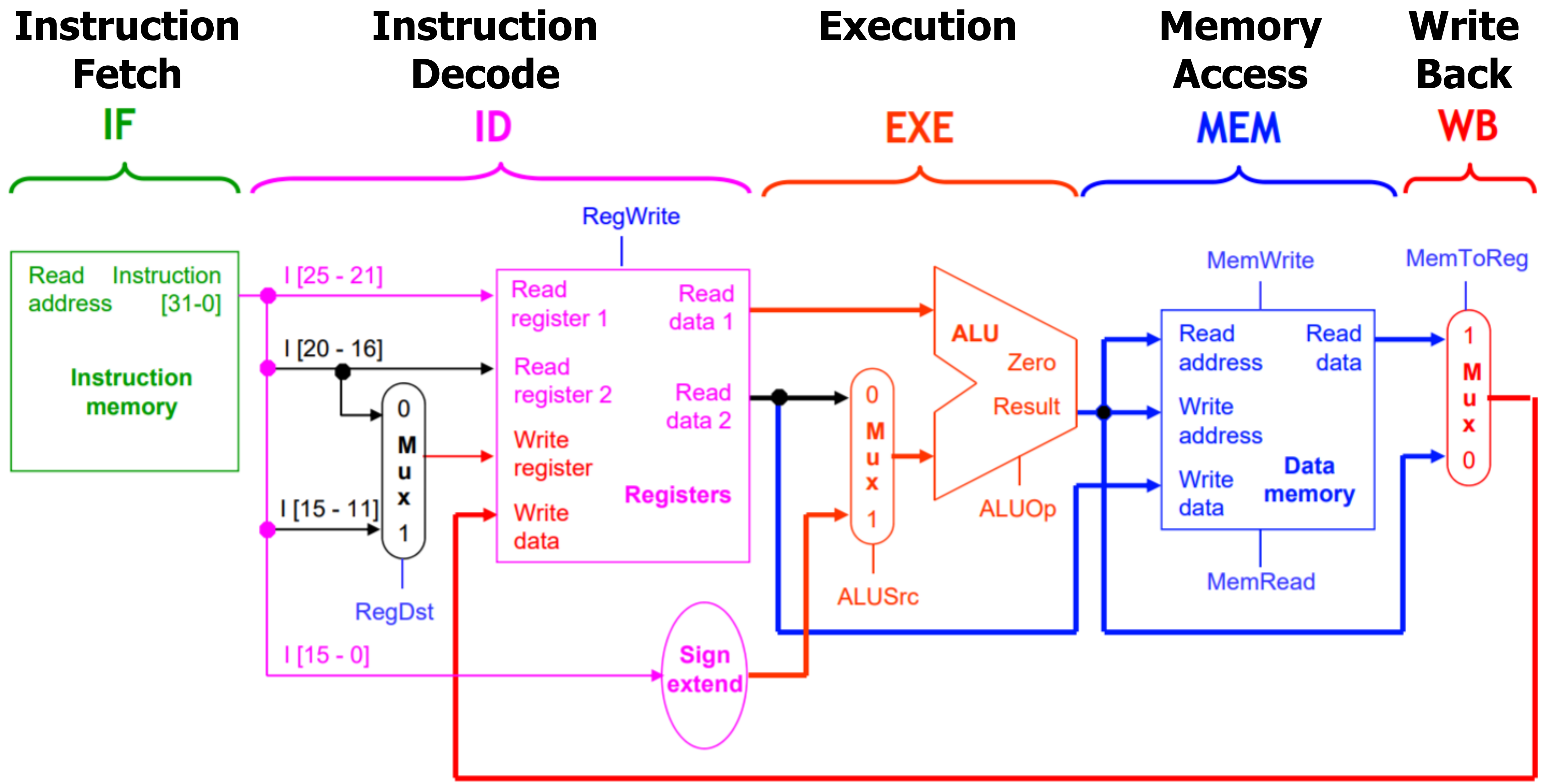
Source: https://www.allaboutcircuits.com/projects/how-to-build-your-own-discrete-4-bit-alu/

Simple CPU with Single Cycle Datapath





Five Stages of Datapath

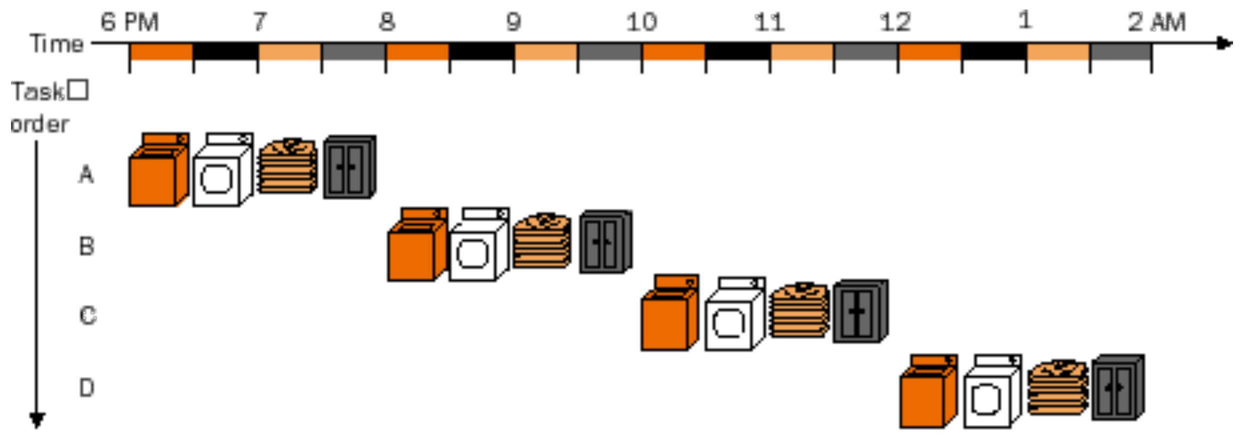


Five Stages of Instruction Execution

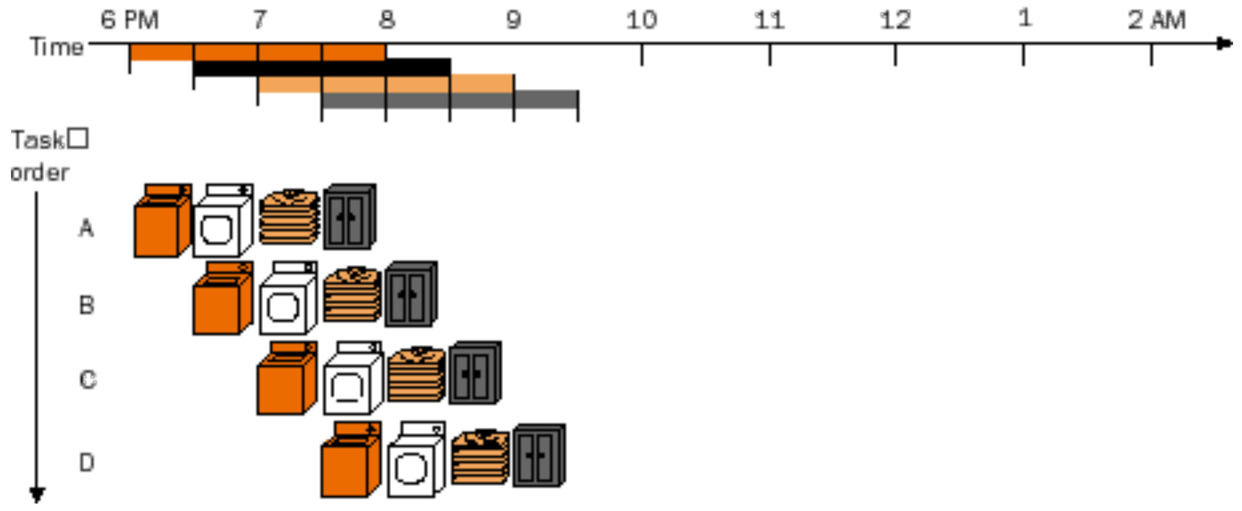
- IF (Instruction Fetch)
  - Reads the instruction at PC
  - PC += 4 (assuming 4-byte RISC instructions)
- ID (Instruction Decode)
  - Understands the instruction
  - Reads registers
- EX (Execute)
  - Performs the operation
  - Arithmetic/Logical Operations
- MEM (Memory Access)
  - Loads from memory or
  - Stores to memory
- WB (Write Back)
  - Writes the result to appropriate registers

Laundry Pipeline Example

No pipelining: only 1 worker



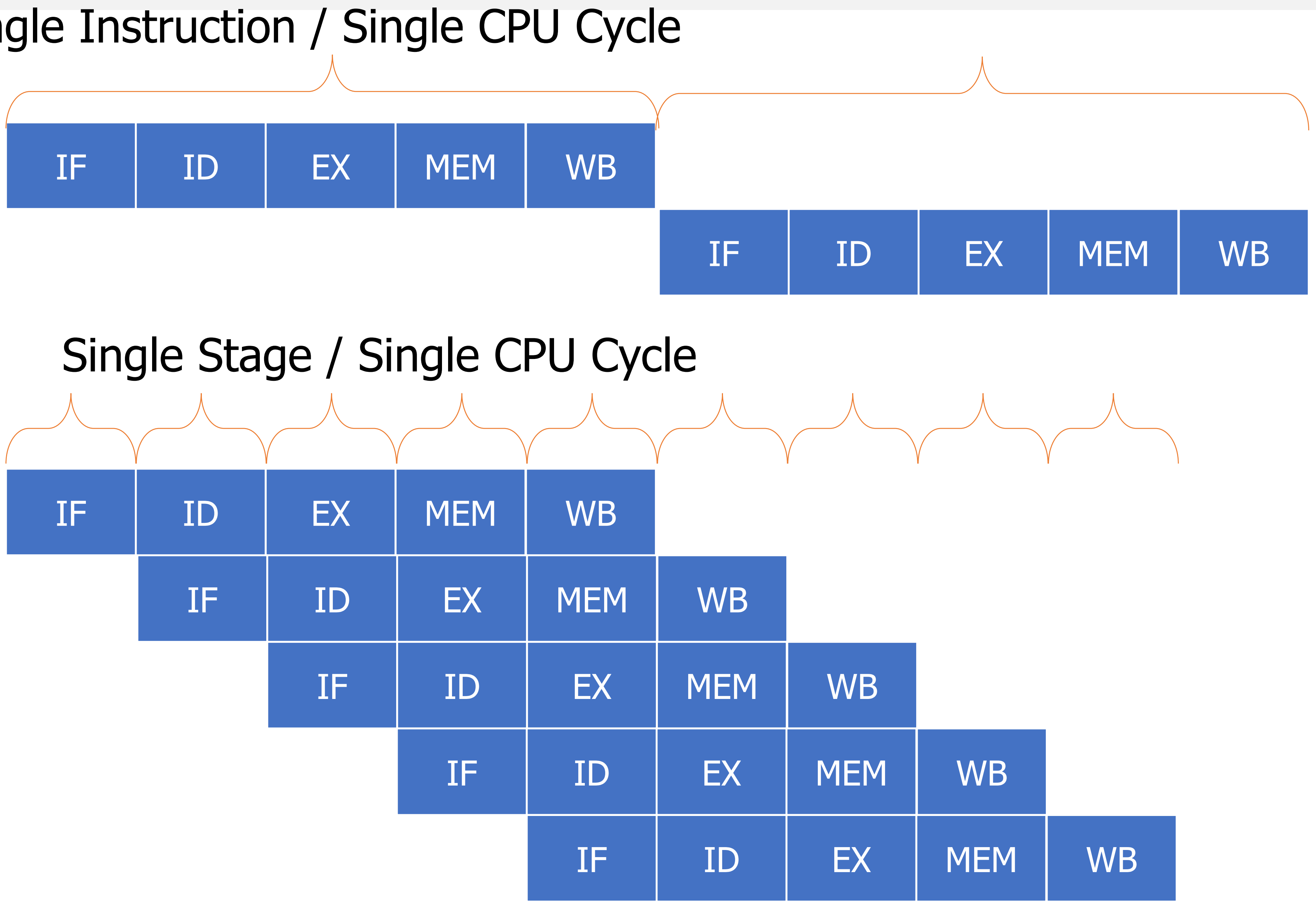
With pipelining: 4 workers



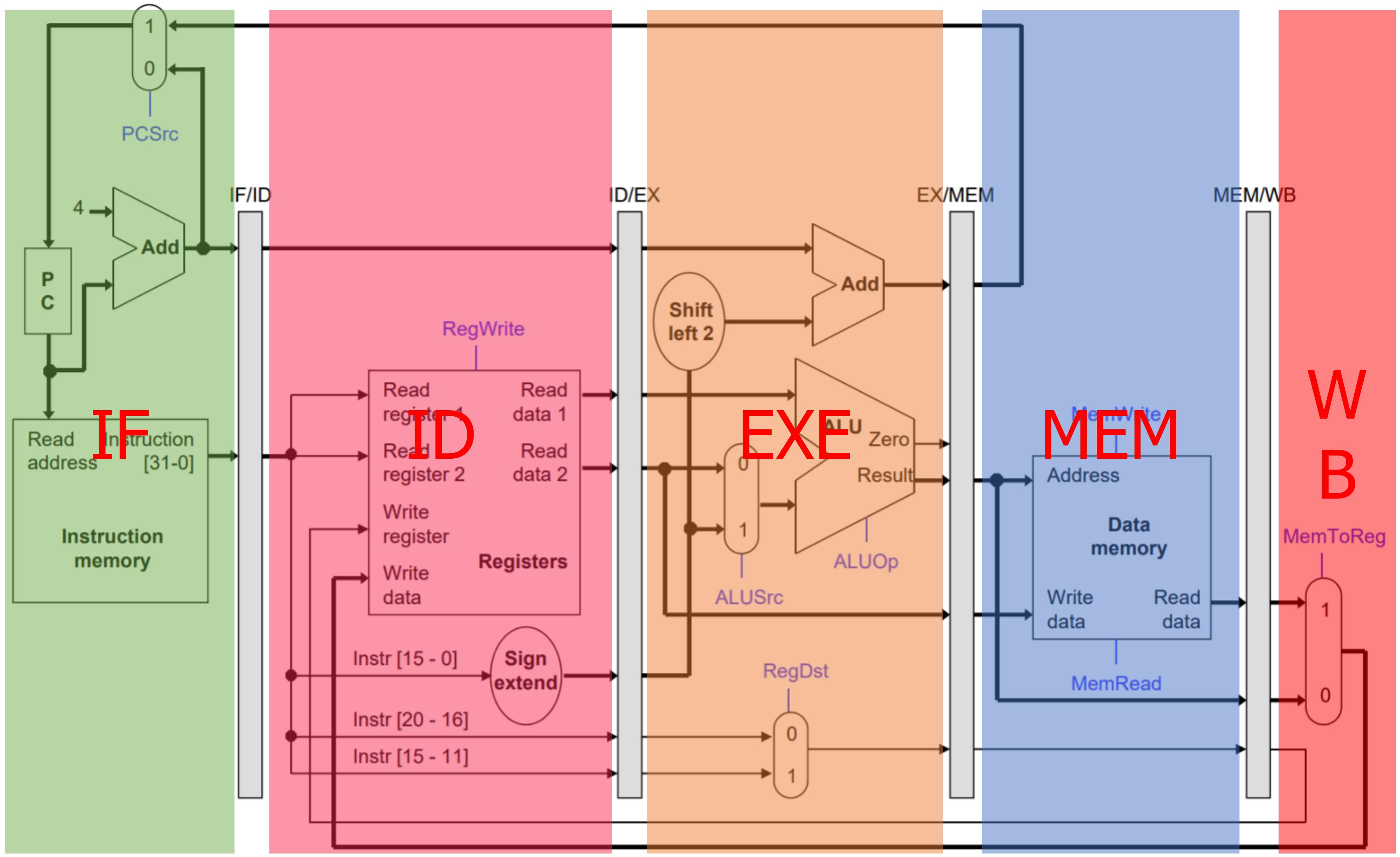
Pipelining

Sequential Execution

(Ideal) Pipelined Execution

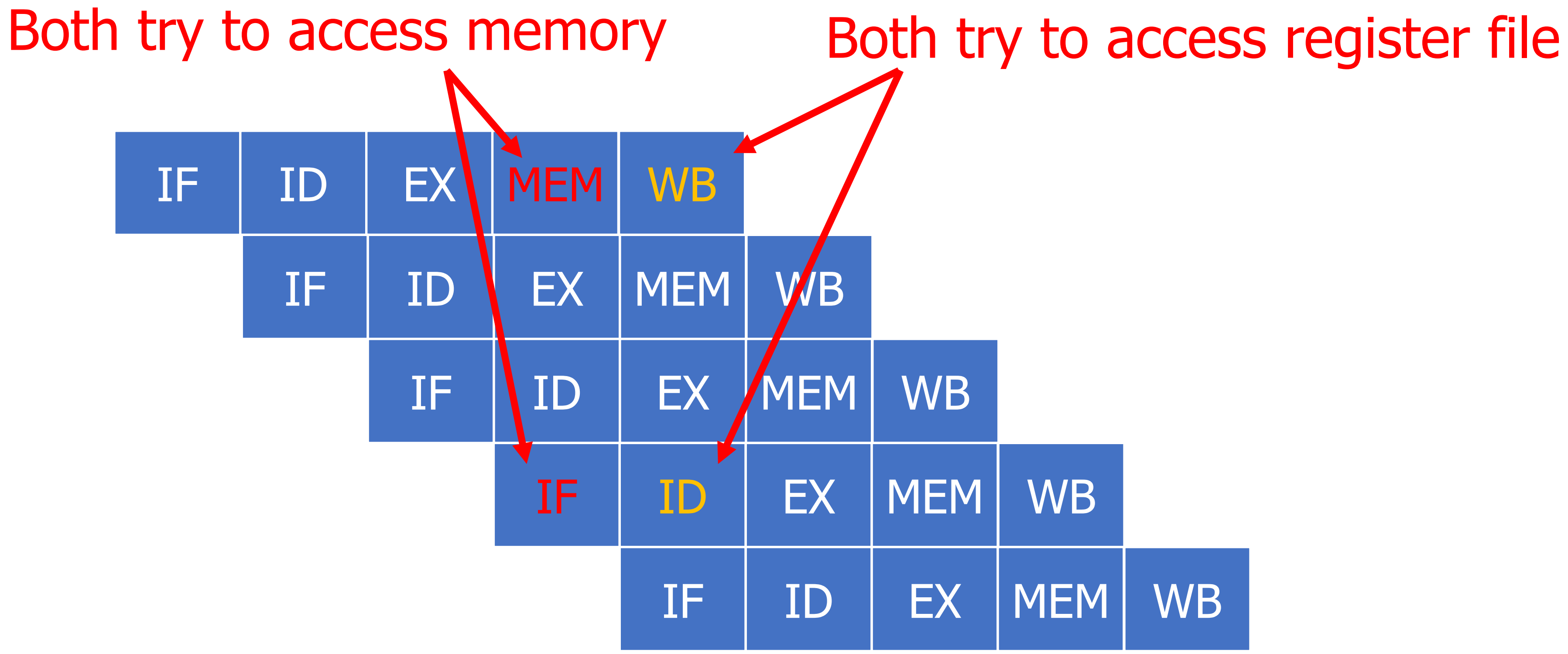


Pipelined Datapath



Pipeline Hazards

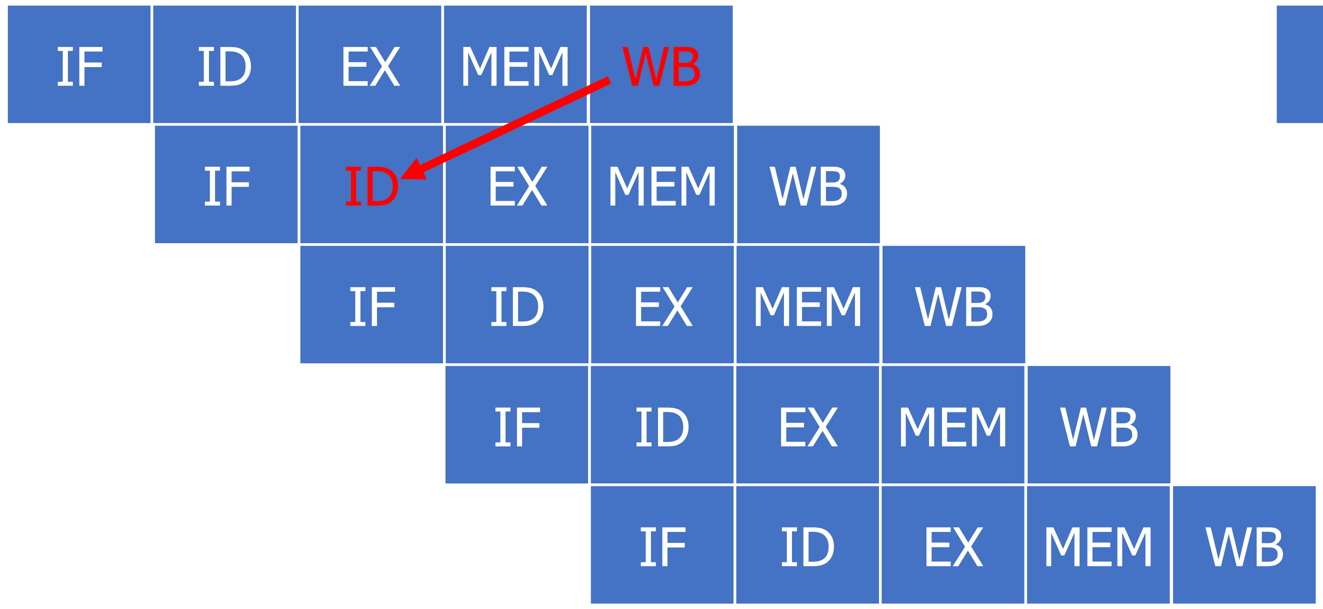
- Structural Hazards
  - HW resource conflicts
  - Harvard architecture is better in terms of pipelining

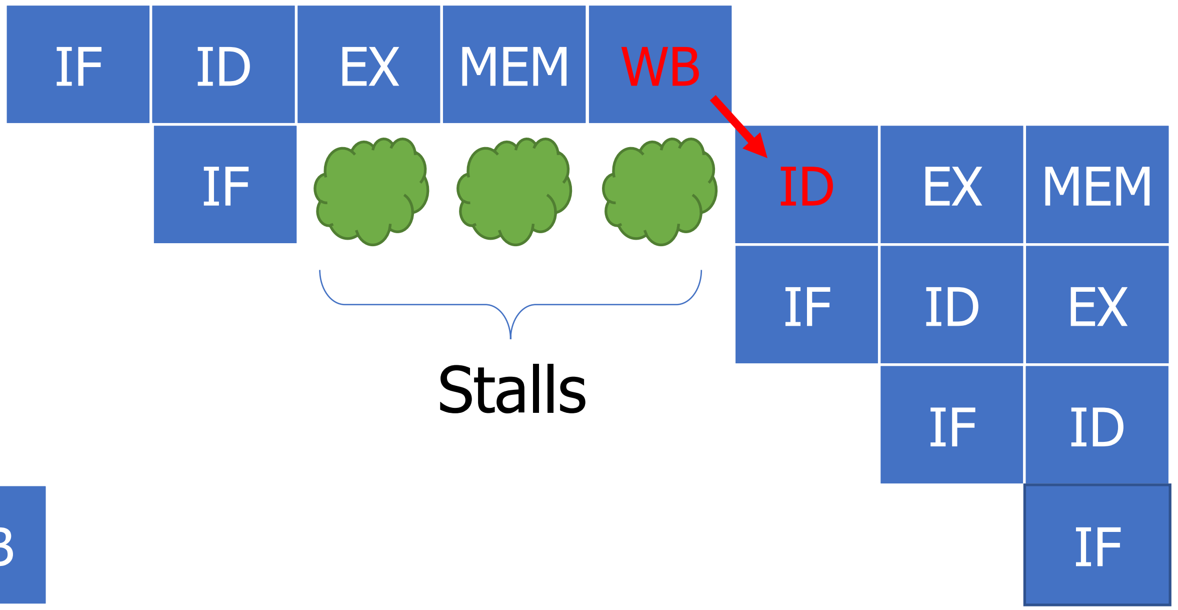


Pipeline Hazards

- Data Hazards
  - Data dependencies
  - RAW, WAR, WAW

```
add         r3, r2, r3 # r3 = r2 + r3
mov         r4, r3     # r4 = r3
```





Pipeline Hazards

- Control Hazards
  - Control uncertainty
  - Conditional branch

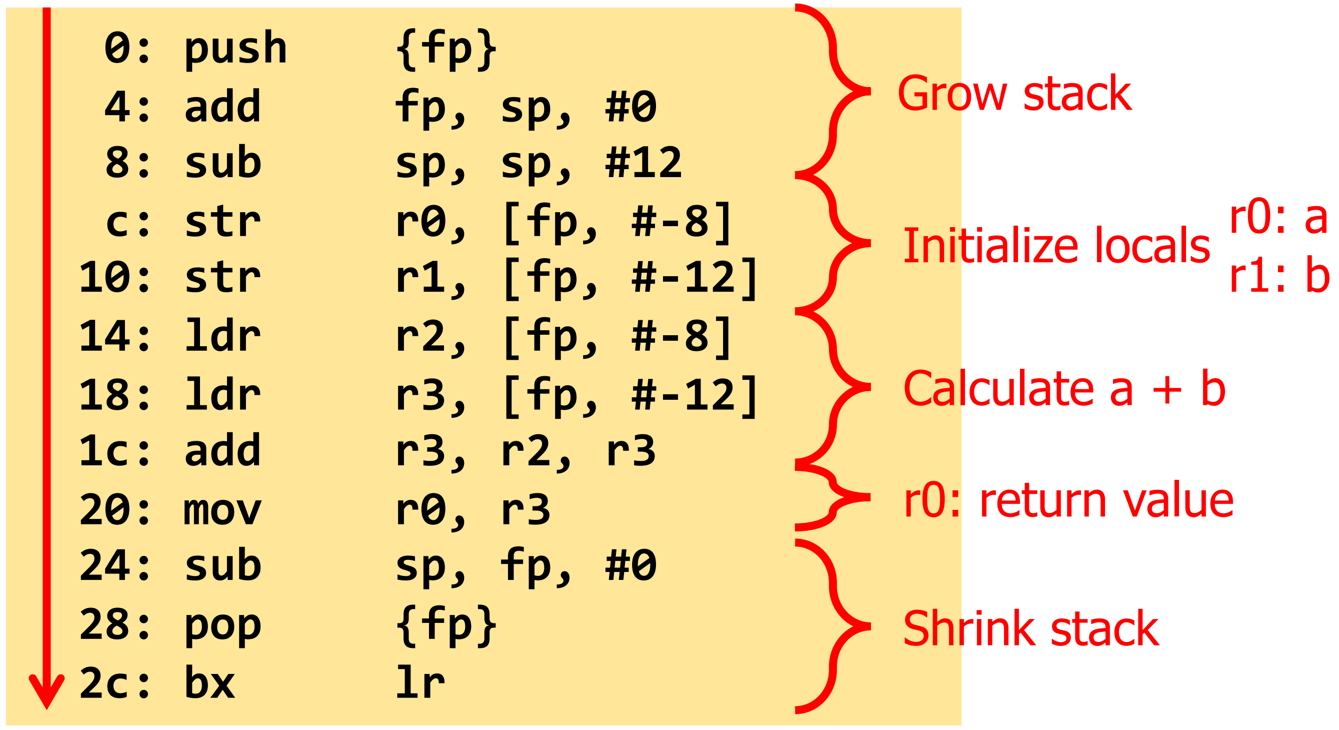
```
int sum(int a, int b)
{
    return a + b;
}
```

- No control uncertainty

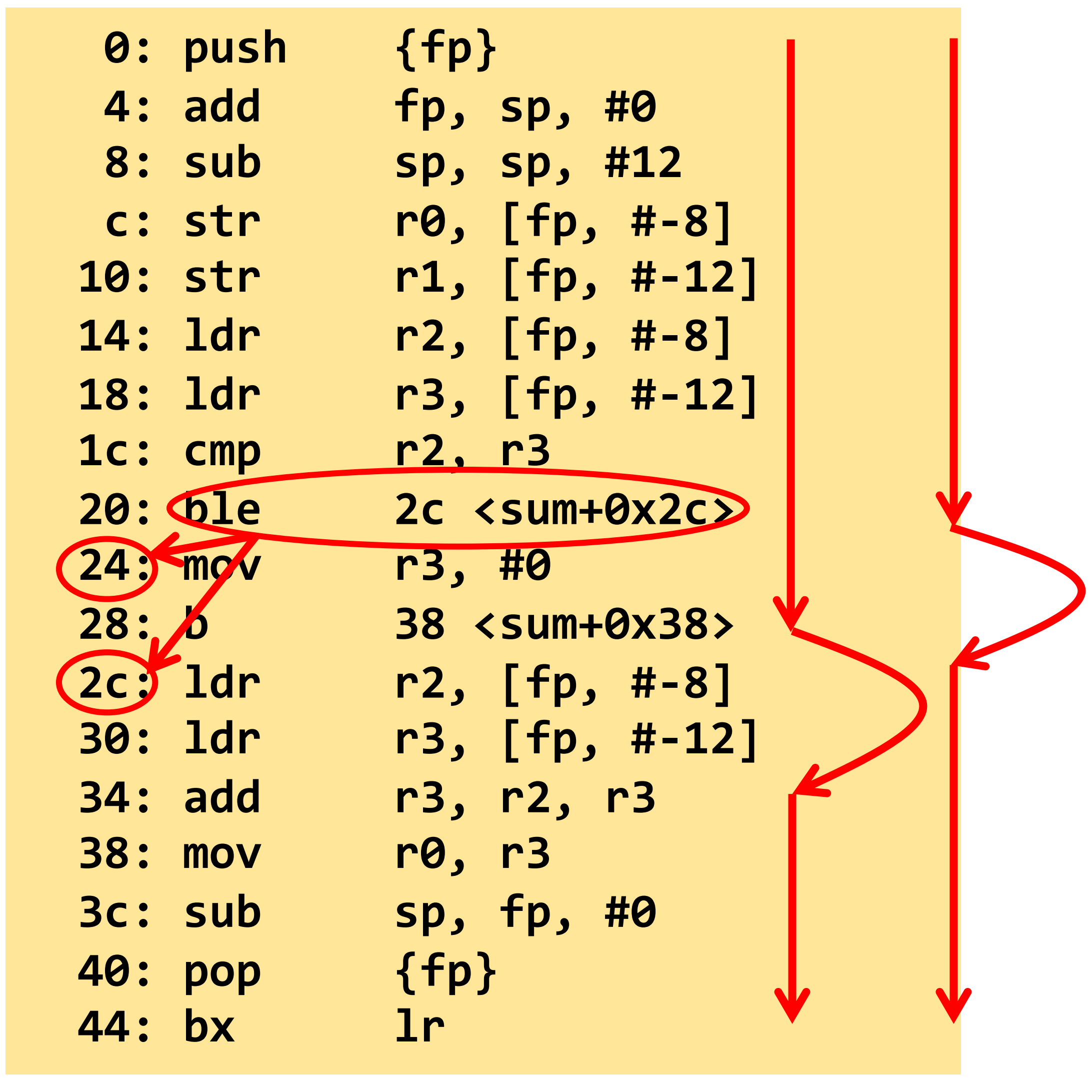
```
int sum(int a, int b)
{
    if (a > b) {
        return 0;
    }
    else {
        return a + b;
    }
}
```

- Control uncertainty
  - if or else

Control Uncertainty



```
 0: push    {fp}
 4: add     fp, sp, #0
 8: sub     sp, sp, #12
 c: str     r0, [fp, #-8]
10: str     r1, [fp, #-12]
14: ldr     r2, [fp, #-8]
18: ldr     r3, [fp, #-12]
1c: add     r3, r2, r3
20: mov     r0, r3
24: sub     sp, fp, #0
28: pop     {fp}
2c: bx      lr
```



```
 0: push    {fp}
 4: add     fp, sp, #0
 8: sub     sp, sp, #12
 c: str     r0, [fp, #-8]
10: str     r1, [fp, #-12]
14: ldr     r2, [fp, #-8]
18: ldr     r3, [fp, #-12]
1c: cmp     r2, r3
20: ble     2c <sum+0x2c>
24: mov     r3, #0
28: b       38 <sum+0x38>
2c: ldr     r2, [fp, #-8]
30: ldr     r3, [fp, #-12]
34: add     r3, r2, r3
38: mov     r0, r3
3c: sub     sp, fp, #0
40: pop     {fp}
44: bx      lr
```

*. ble: branch less than or equal

Speculation and Branch Prediction

- Speculative execution
  - Doing (possibly) useless is better than doing nothing
  - In case of incorrect prediction, flush the pipeline and begin from the start

- Branch prediction
  - Improves the probability of taking the right branch
  - Based on the historical data

Out-Of-Order Execution

- Changes the execution order of instructions at runtime

```
1: A = B + C;
2: E = A + F;
3: D = B + 1;
```

- Line 2 depends on Line 1’s result
- High probability of pipeline stalls

```
1: A = B + C;
2: D = B + 1;
3: E = A + F;
```

- Reordered instructions
- Less probability of pipeline stalls
- Produces the same result as the original execution order

Processor Performance Metrics

- Latency (execution time)
  - Time to finish a program
  - A program's perspective
- Throughput (bandwidth)
  - Number of programs processed in a certain time
  - A system's perspective

Taxi is better for latency
Bus is better for throughput

Image Source: https://cognigen-cellular.com/explore/taxi-clipart-taxi-bus/

Execution Time

- Seconds / cycle
  - Higher CPU clock frequency (no longer possible)
  - Energy consumption and heat problem
  - Multicore CPU can provide more CPU cycles
- Cycles / instruction
  - CISC CPU takes more CPU cycles per instruction (RISC wins)
- Instructions / program
  - RISC compilers produce more instructions for a given program (CISC wins)

$$\frac{\text{seconds}}{\text{program}} = \frac{\text{seconds}}{\text{cycle}} \times \frac{\text{cycles}}{\text{instruction}} \times \frac{\text{instructions}}{\text{program}}$$

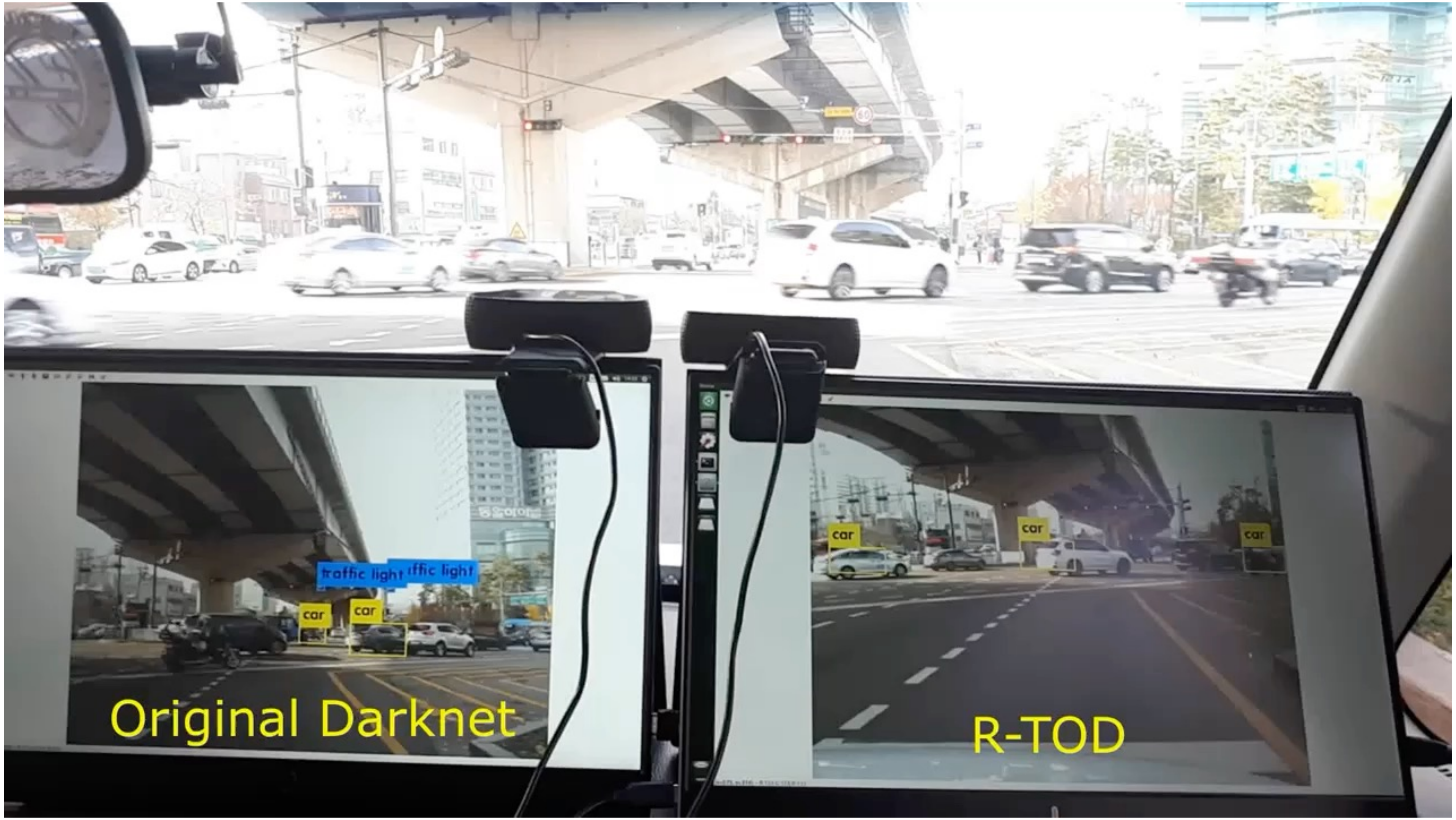
Throughput

- MIPS (Million Instructions Per Second)
  - How many instructions can a CPU execute in a second
  - Historically important but not that useful these days

$$\text{MIPS} = \frac{\text{instructions}}{\text{cycles}} \times \frac{\text{cycles}}{\text{second}}$$

Will high throughput always lead to low latency?

Same Throughput, Different Latencies



Summary

- Pipeline Architecture
- Pipeline Hazards
  - Structural Hazards
  - Data Hazards
  - Control Hazards
- Performance Metrics
  - Latency
  - Throughput