VE370 RC (Midterm)
Pipeline & Data hazards

Li Shi

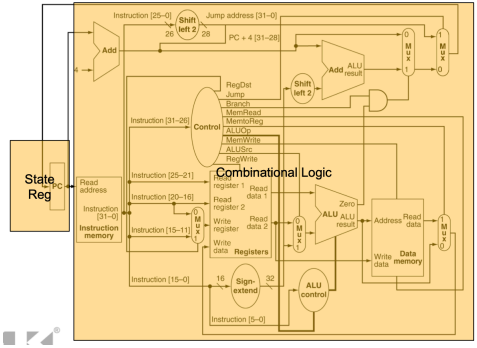
2020.10.17

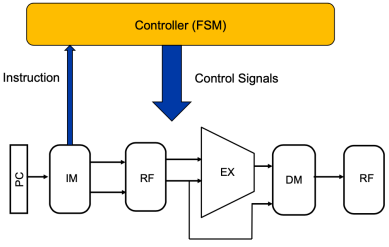
Key concept 1: 3 types of microarchitecture

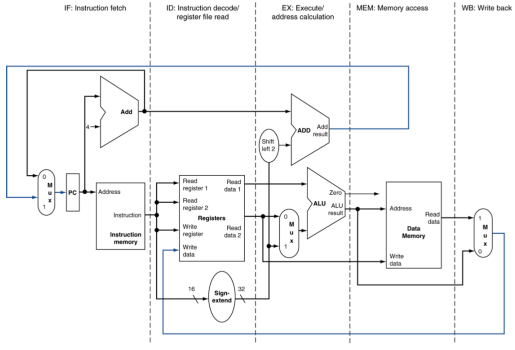
- Single cycle

Multi cycle

Pipeline



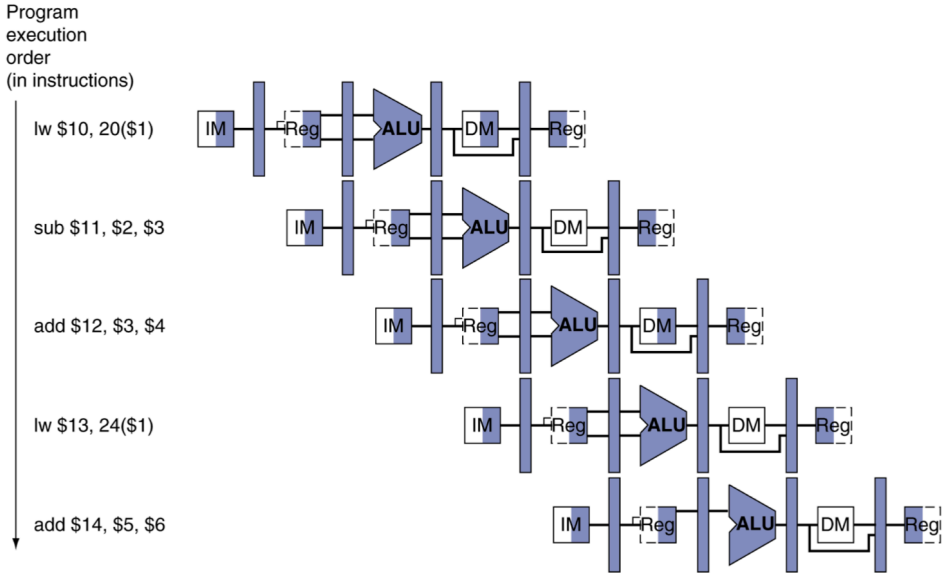


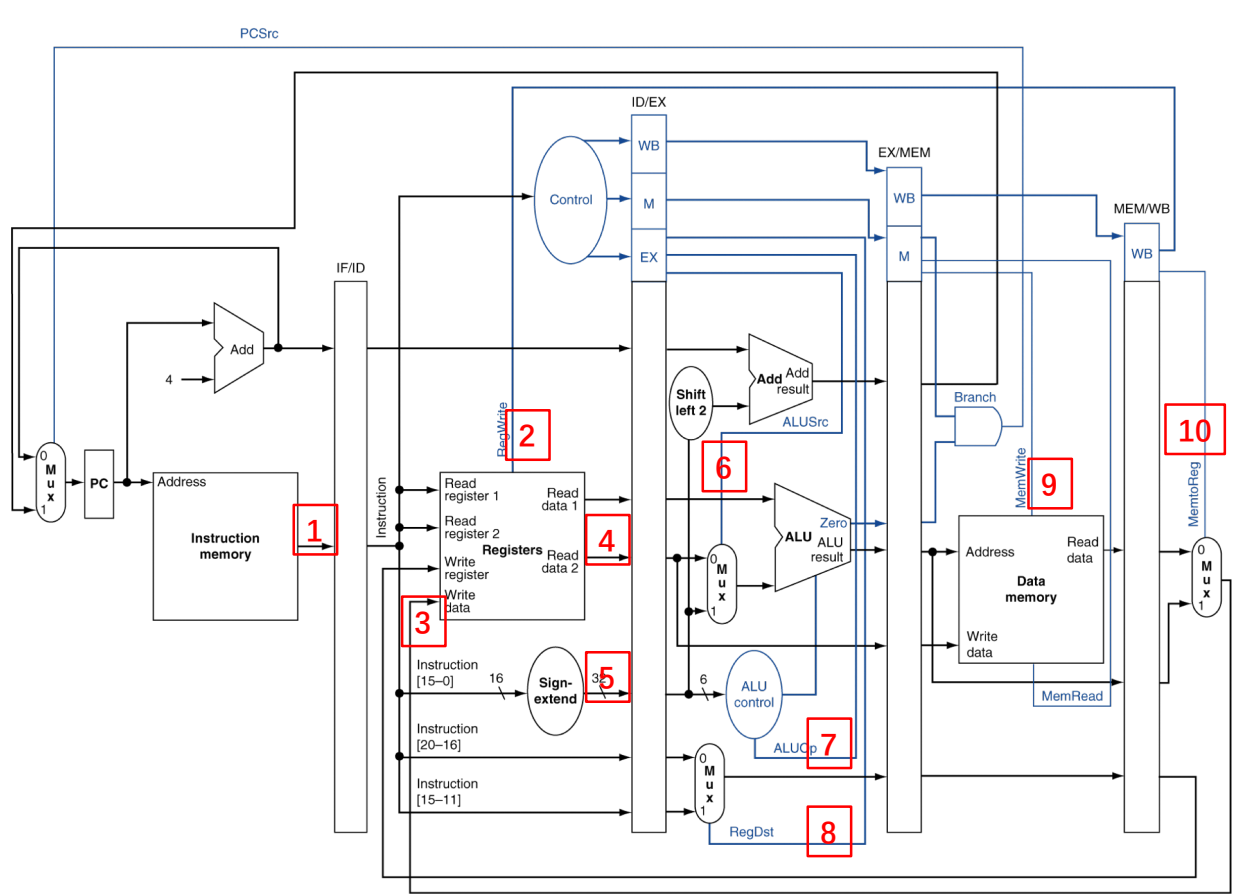


Microarchitecture	CPI	cycle time
Single-cycle unpipelined	1	long
Pipelined	1	short
Multi-cycle, unpipelined control	>1	short

Class exercise 1: Signals in pipelined CPU

- We have 5 instructions in the pipeline.
- Currently the first `lw` instruction is at WB stage.
- Find the values on the following wires.
- (See next slide)






```
WB:  lw  $10, 20($1)
MEM: sub $11, $2, $3
EX:  add $12, $3, $4
ID:  lw  $13, 24($1)
IF:  add $14, $5, $6
```

1:

2:

3:

4:

5:

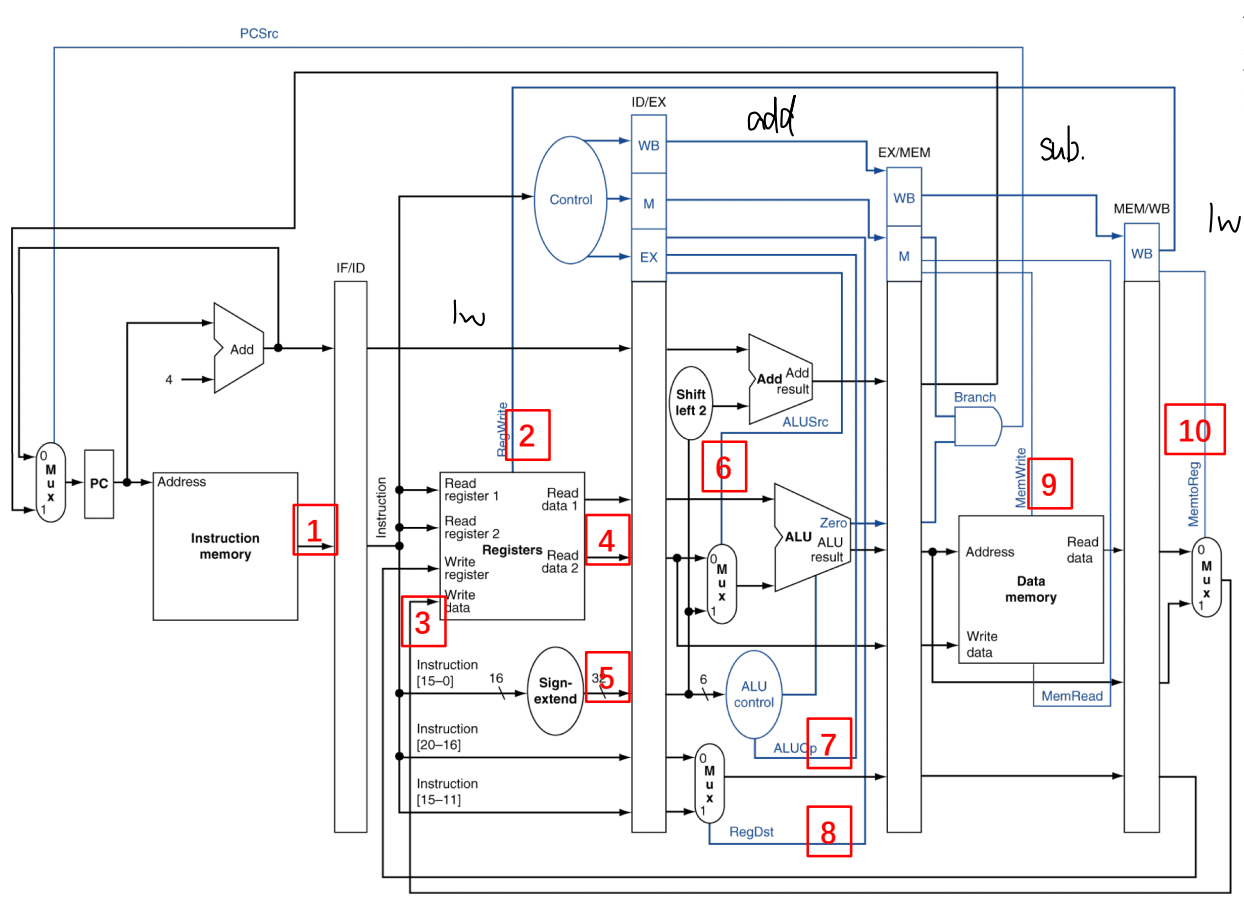
6:

7:

8:

9:

10:




```
WB:  lw  $10, 20($1)
MEM: sub $11, $2, $3
EX:  add $12, $3, $4
ID:  lw  $13, 24($1)
IF:  add $14, $5, $6
```

```
1:   000000 00101 00110
     01110 00000 100000
2:   1
3:   Mem[Reg[$1]+20]
4:   Reg[$0]
5:   0000000000000000
(24) 0000000000011000
6:   0
7:   10
8:   1
9:   0
10:  0
```

Key concept 2: 3 types of hazards

- Structural hazards
  - `lw $1, 0($2)` and `lw $3, 0($4)`
  - Execute 2 instructions at the same time (so-called superscalar processor), but there is only one read/write port in data memory
  - Simple 5-stage MIPS pipeline has no structural hazards
- Data hazards
  - `add $1, $2, $3`
  - `add $4, $1, $5`
- Control hazards (not covered so far)
  - `add $1, $2, $3`
  - `beq $4, $5, …`
  - `add $6, $7, $8`

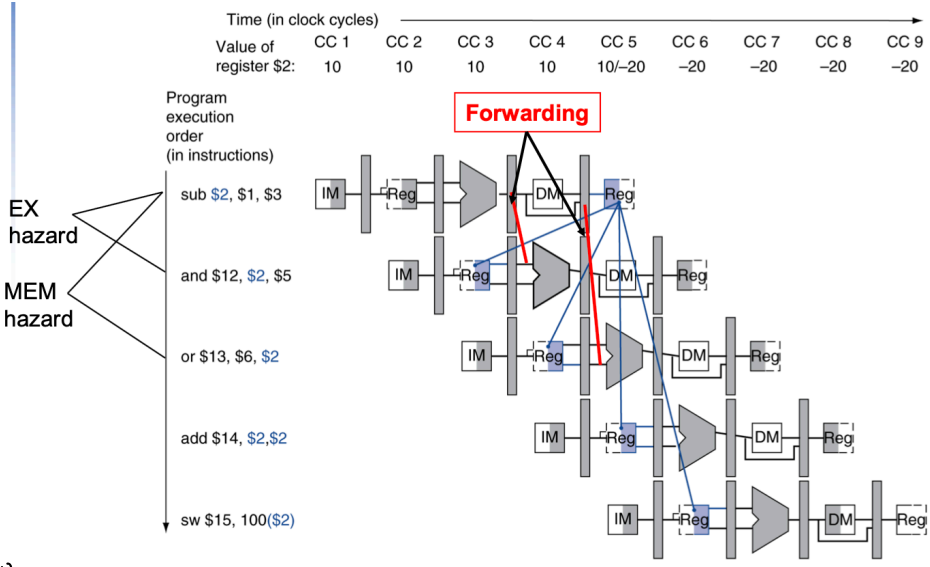
Key concept 3: Data hazard

- Data hazards occur when one instruction depends on a data value produced by a preceding instruction still in the pipeline
- Approaches to resolving data hazards
  - **Schedule:** Programmer explicitly avoids scheduling instructions that would create data hazards
  - **Stall:** Hardware includes control logic that freezes earlier stages until preceding instruction has finished producing data value
  - **Bypass:** Hardware datapath allows values to be sent to an earlier stage before preceding instruction has left the pipeline

Key concept 3: Data hazard

- Two kinds of data hazard
  - Solvable by Bypassing or Forwarding Unit
  - Unsolvable, and insert nops
- (We can also avoid hazards and stalls by arranging the code when compiling)

nop 是forwards 无法解决的.

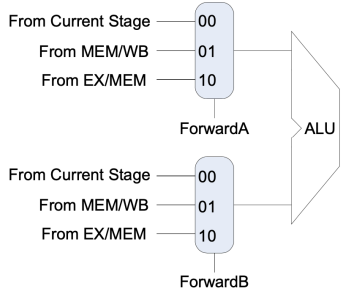


Key concept 4: Forwarding unit

- 2 important formulas (No need to memorize. Try to understand!)

EX hazard 1 & 2 hazard

- if (EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0)
  and (EX/MEM.RegisterRd == ID/EX.RegisterRs))
  MUX select signal ForwardA = 10
- if (EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0)
  and (EX/MEM.RegisterRd == ID/EX.RegisterRt))
  MUX select signal ForwardB = 10



MEM hazard (not EX hazard) 1 & 3 hazard

- if (MEM/WB.RegWrite and (MEM/WB.RegisterRd ≠ 0)
  and (MEM/WB.RegisterRd = ID/EX.RegisterRs)
  and not (EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0)
  and (EX/MEM.RegisterRd = ID/EX.RegisterRs)))
  ForwardA = 01
- if (MEM/WB.RegWrite and (MEM/WB.RegisterRd ≠ 0)
  and (MEM/WB.RegisterRd = ID/EX.RegisterRt)
  and not (EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0)
  and (EX/MEM.RegisterRd = ID/EX.RegisterRt)))
  ForwardB = 01

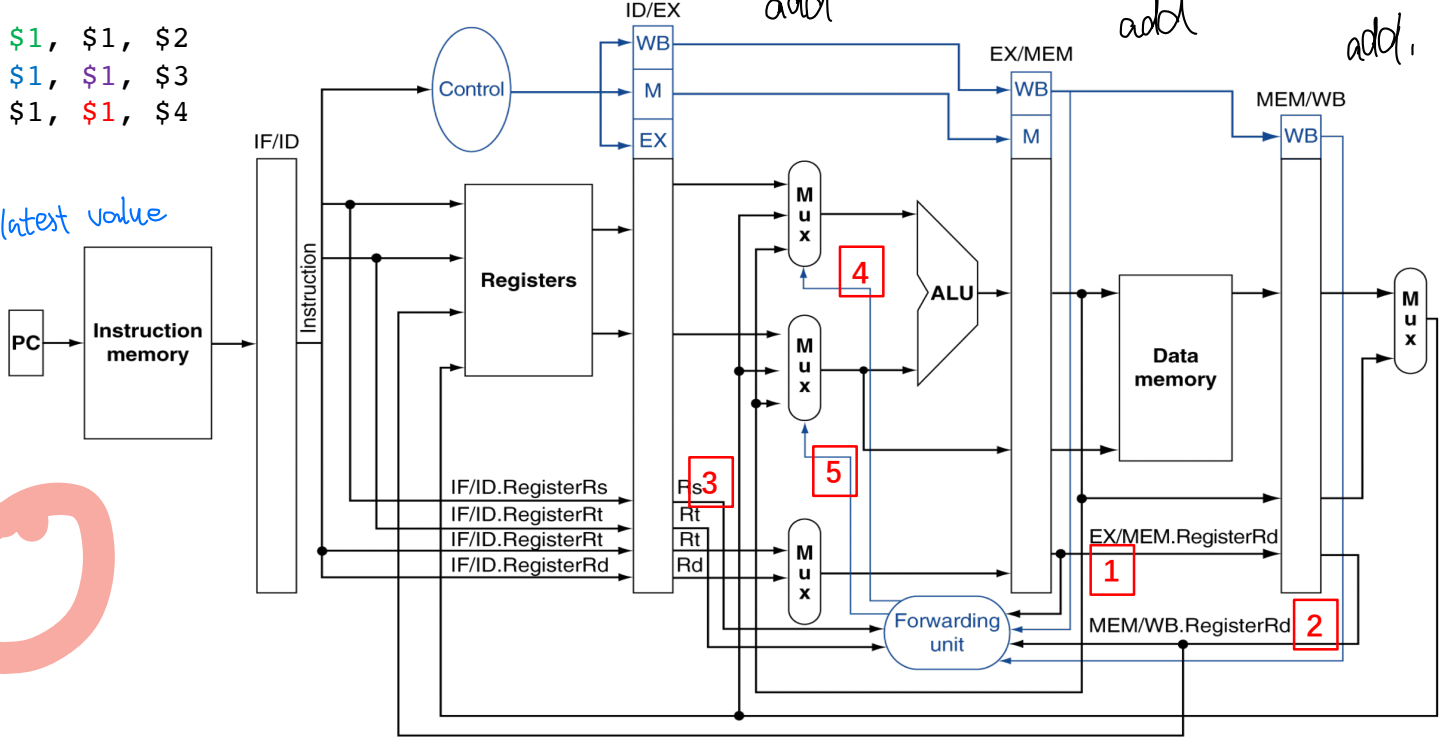
Class exercise 2: Signals in forwarding unit

- We have 5 instructions in the pipeline.
- Currently the first **`add`** instruction is at WB stage.
- Find the values on the following wires.
- (See next slide)

```
WB:  add $1, $1, $2
MEM: add $1, $1, $3
EX:  add $1, $1, $4
ID:  nop
IF:  nop
```

```
WB:  add $1, $1, $2
MEM: add $1, $1, $3
EX:  add $1, $1, $4
ID:  nop
IF:  nop
```

choose latest value

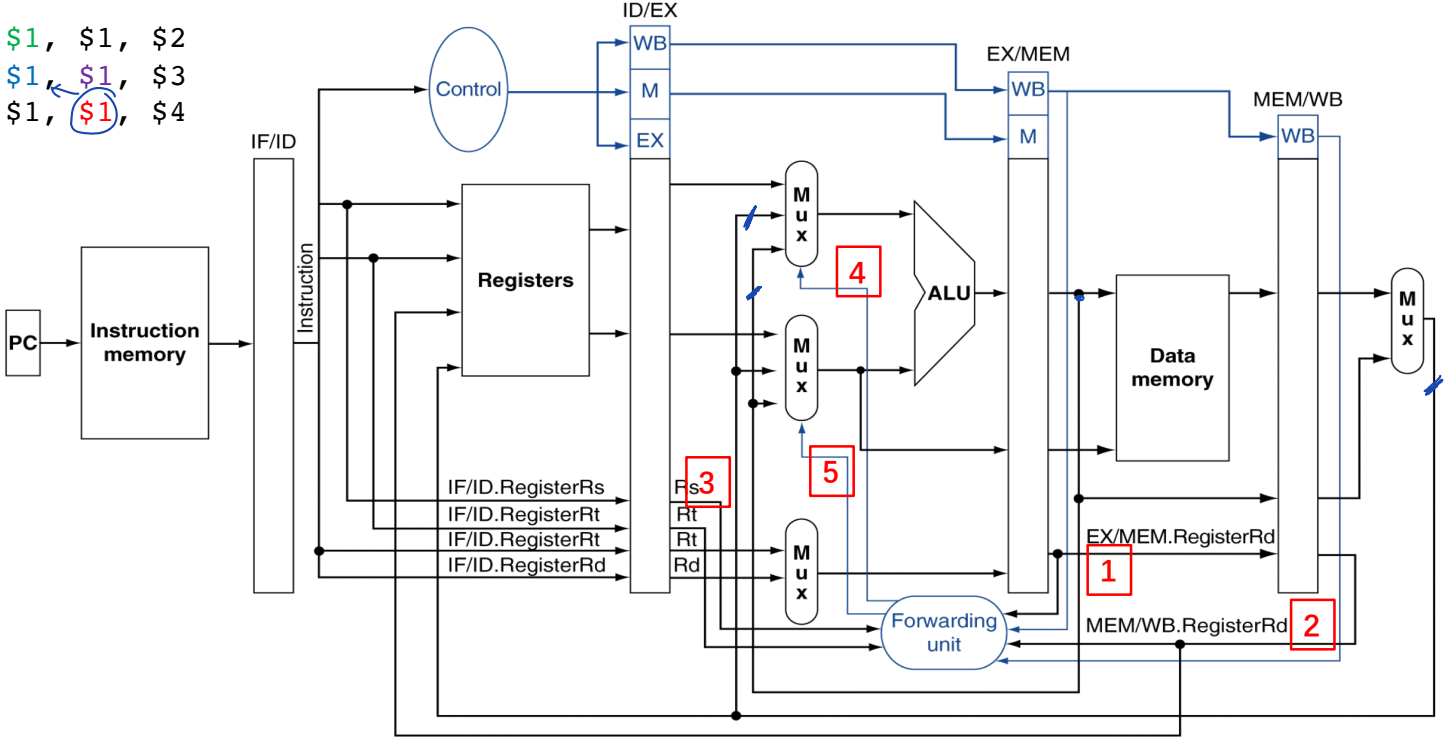


EX/MEM.RegWrite: 1: 1 4: 10

MEM/WB.RegWrite: 2: 1 5: 00.

3: 1

```
WB:  add $1, $1, $2
MEM: add $1, $1, $3
EX:  add $1, $1, $4
ID:  nop
IF:  nop
```




```
EX/MEM.RegWrite: 1
MEM/WB.RegWrite: 1
```

1: 00001
2: 00001
3: 00001

4: 01 (From MEM/WB)
5: 00

Notes

- No calculator. No cheating paper.
- Necessary information will be given in the exam.

- If necessary, you could read the textbook by yourself.
  - *Computer Organization and Design: The Hardware/Software Interface*