CS465: Computer Systems Architecture

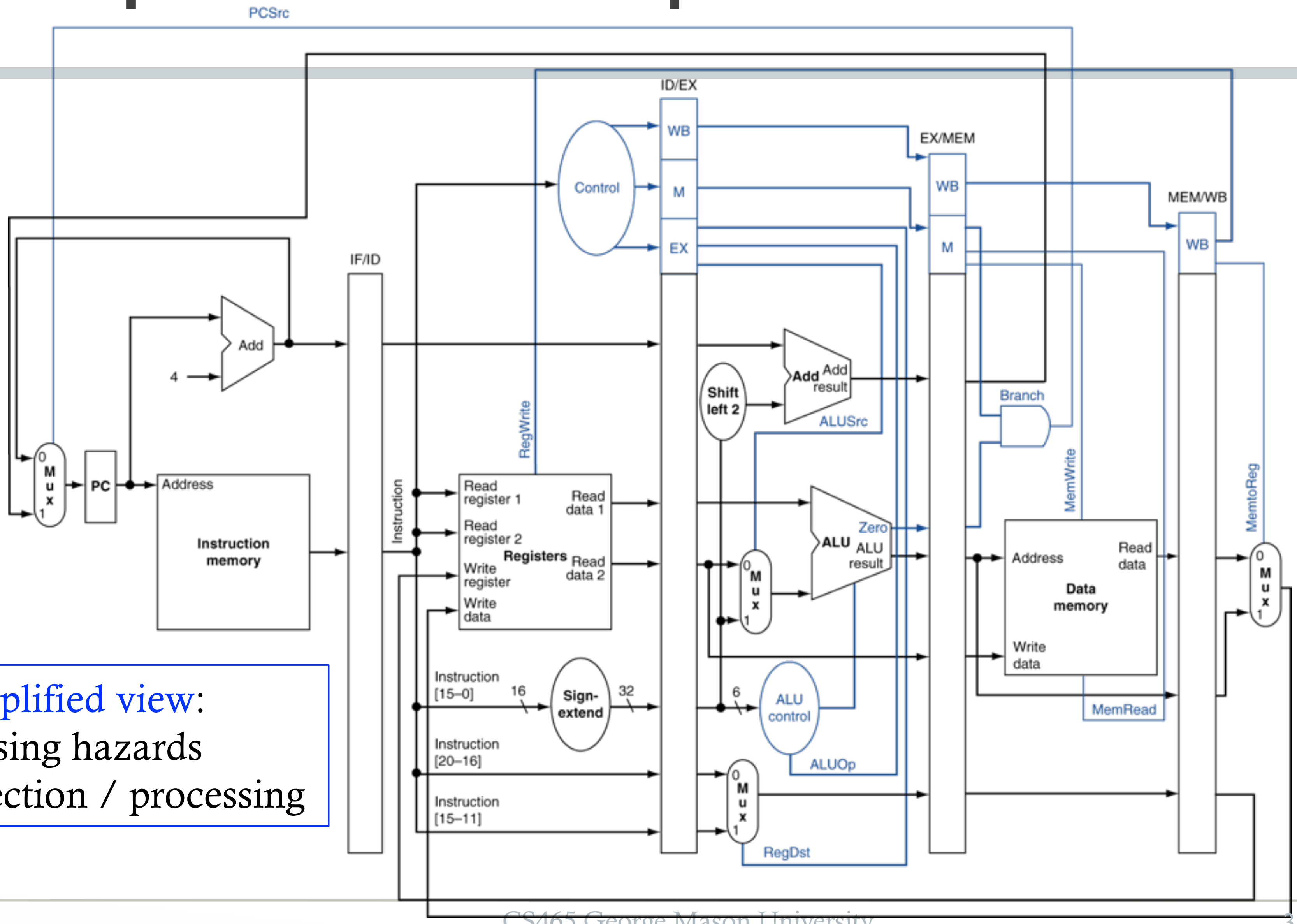
Lecture 8: Processor III – Instruction Level Parallelism

*Slides adapted from Computer Organization and Design by Patterson and Hennessy

Review: Pipelined CPU

- Overlapped execution of multiple instructions
- Each on a different stage using a different major functional unit in datapath
  - IF, ID, EX, MEM, WB
  - Same number of stages for all instruction types
- Improved overall throughput
  - Effective CPI=1 (ideal case)

Pipelined Datapath & Control



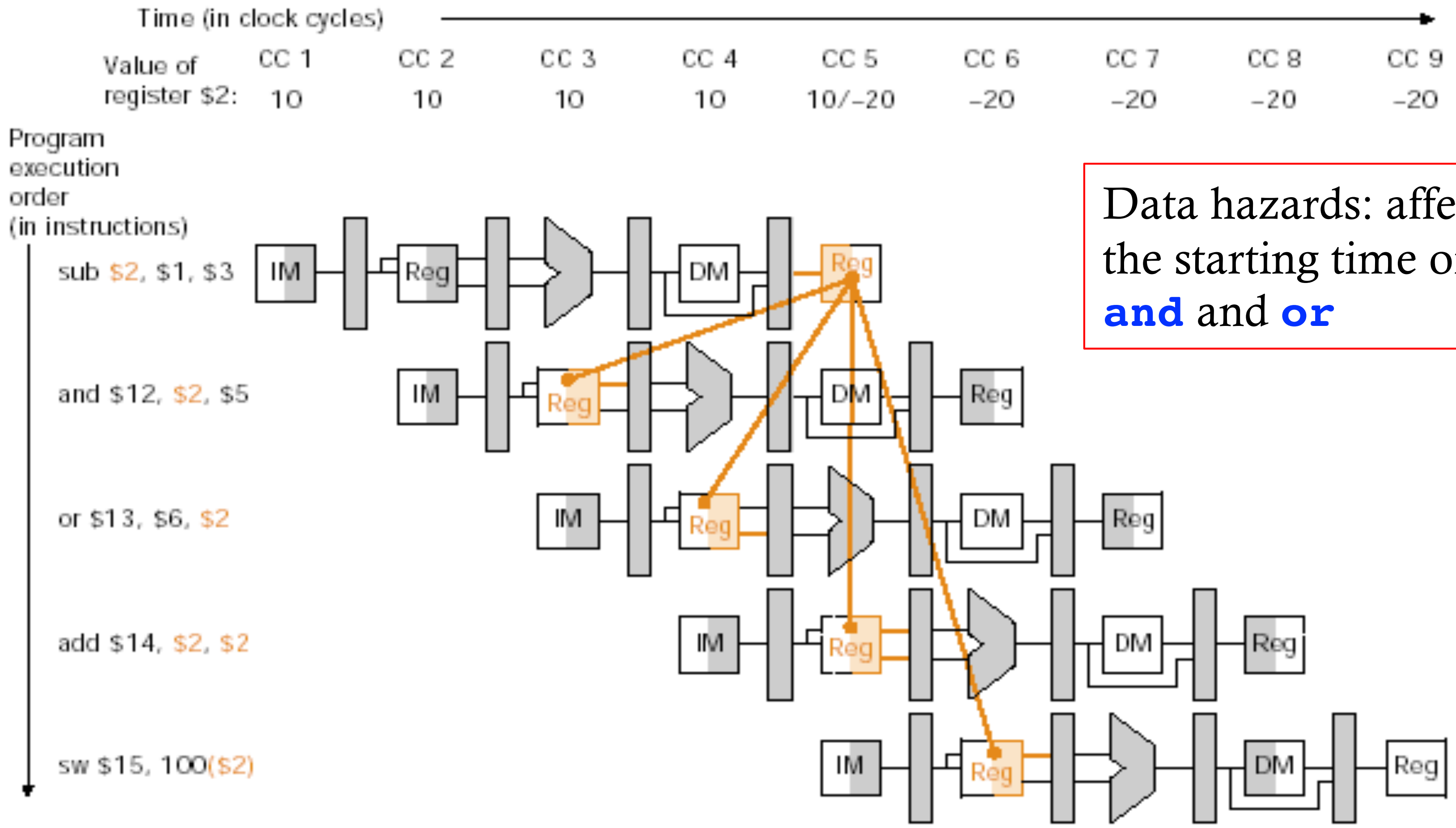
Simplified view: missing hazards detection / processing

Outline

- Resolving hazards
  - Data hazards: forwarding
  - Stalling the pipeline
  - Control hazards; prediction
- Advanced ILP techniques
  - Reordering
  - Static multiple issue
  - Dynamic scheduling and speculation
- Concluding remarks

Data Hazards

- An example: assume initially $2=10, $1=10, $3=30



Data hazards: affect the starting time of **and** and **or**

Data Hazards Detection

- Data hazards occur because
  - Value flows between two instructions in pipeline: true data dependence
  - When the consumer instruction needs the value, the producer instruction has not delivered it
- True data dependence

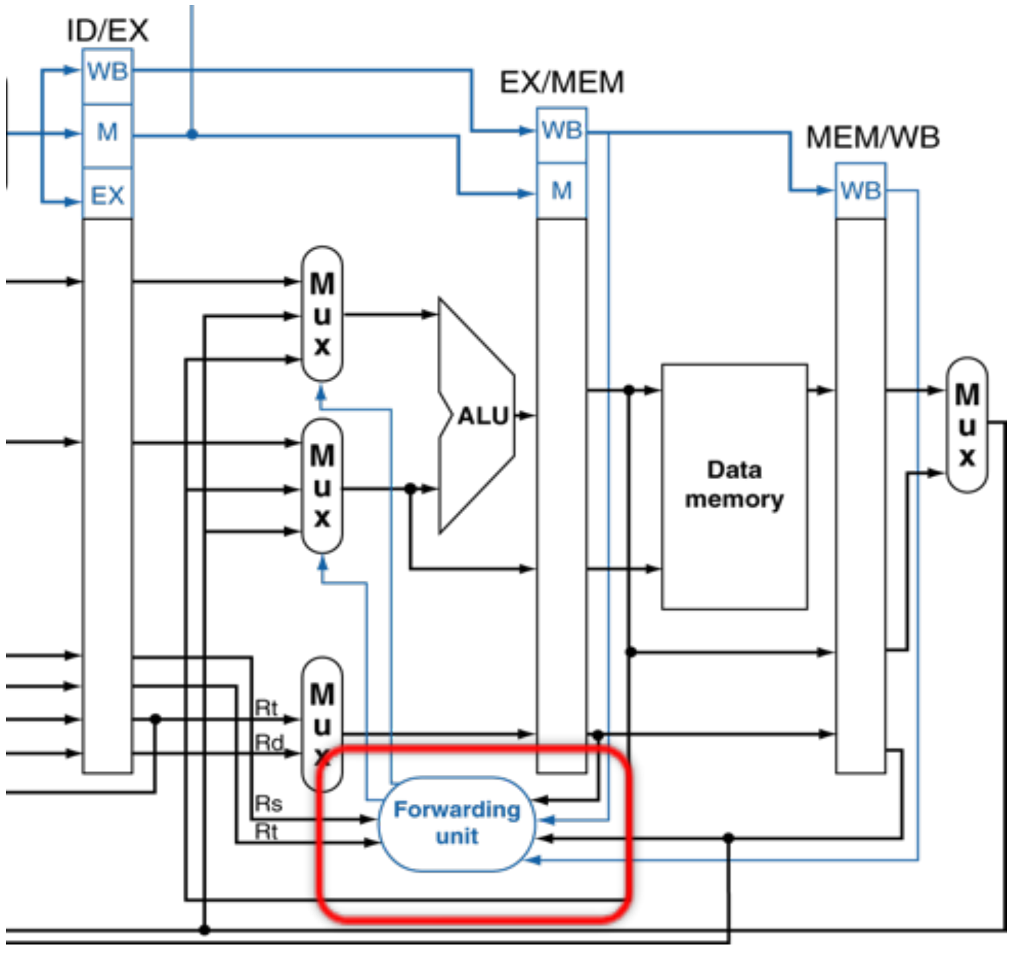
```
x := 4 + a;
…
y := x + 1;
```

  - Also known as flow dependences or read after write hazards

Data Hazards Detection

- True data dependence + bad timing ➔data hazard
- Conditions to check (in pipeline):
  - Shared register between different instructions
    - ➔ We can compare source and destination registers between instructions in pipeline
  - The earlier instruction updates the shared register
    - ➔We can check the RegWrite signal for the earlier instruction
  - Timing: producer of the value has not updated register yet
    - ➔Producer instruction is 1 or 2 cycles before consumer
    - Earlier instructions as producers would not cause any issue

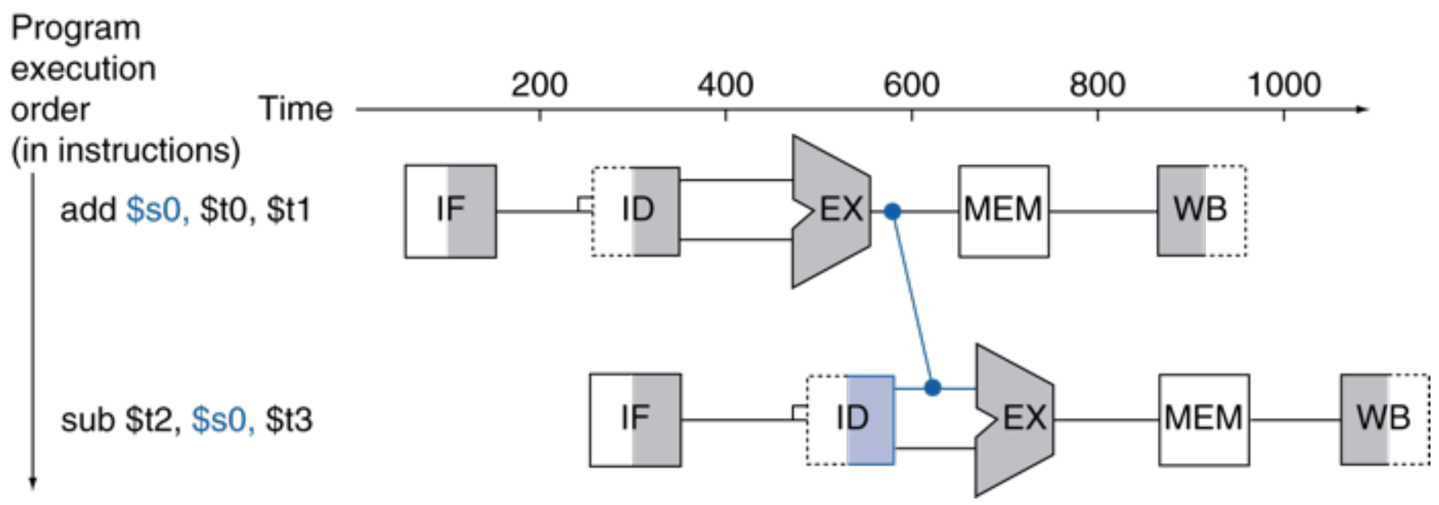
Advanced: Data Hazard Detection



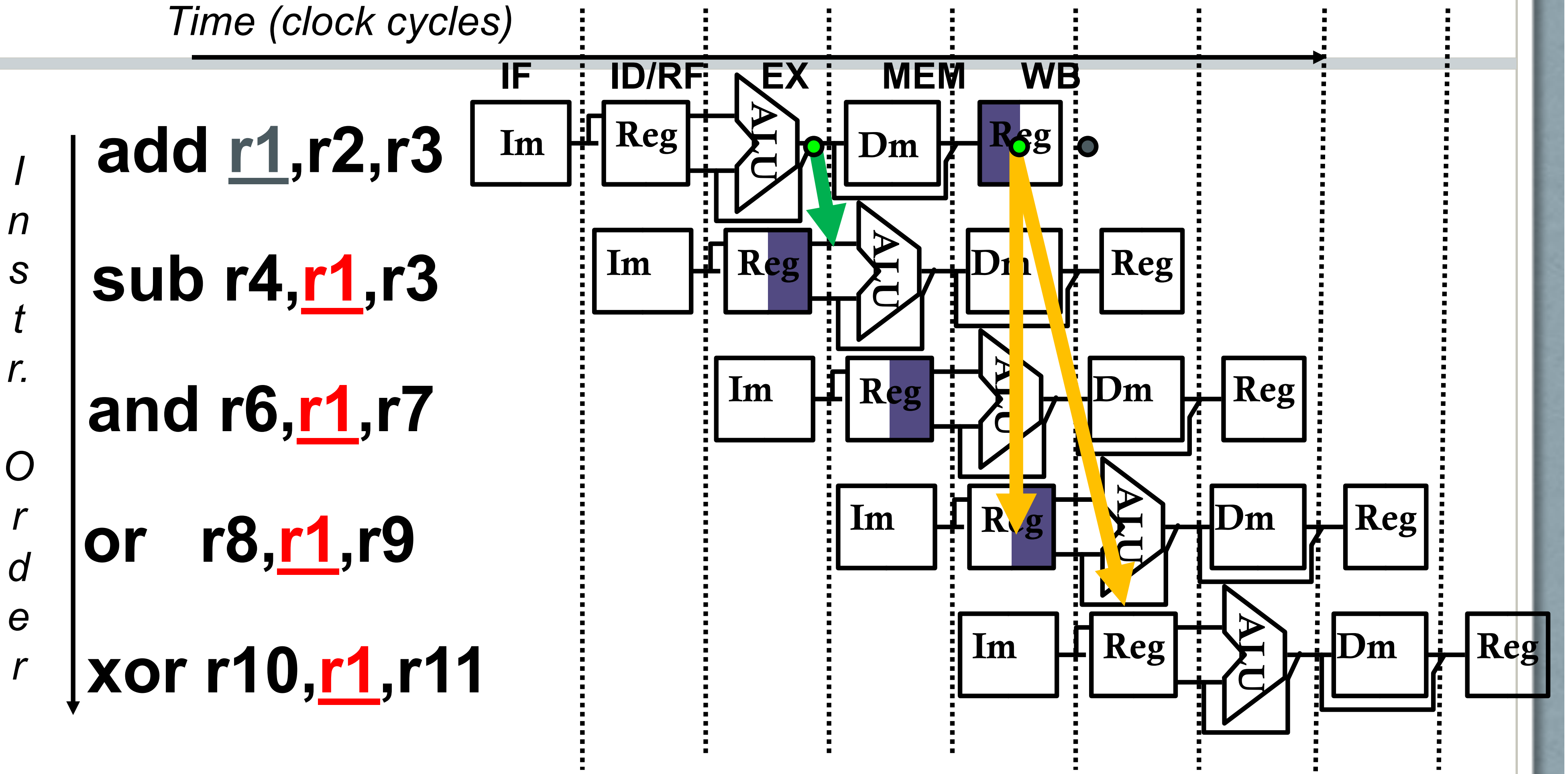
- Consumer: about to use ALU (EX stage)
- Data input:
  - Rs/rt (operands of the later instruction)
  - Destination register of two predecessors in pipeline
- Control signal input:
  - RegWrite of the two predecessors in pipeline

Forwarding (aka Bypassing)

- Use result when it is computed
  - Don't wait for it to be stored in a register
  - Requires extra connections in the datapath

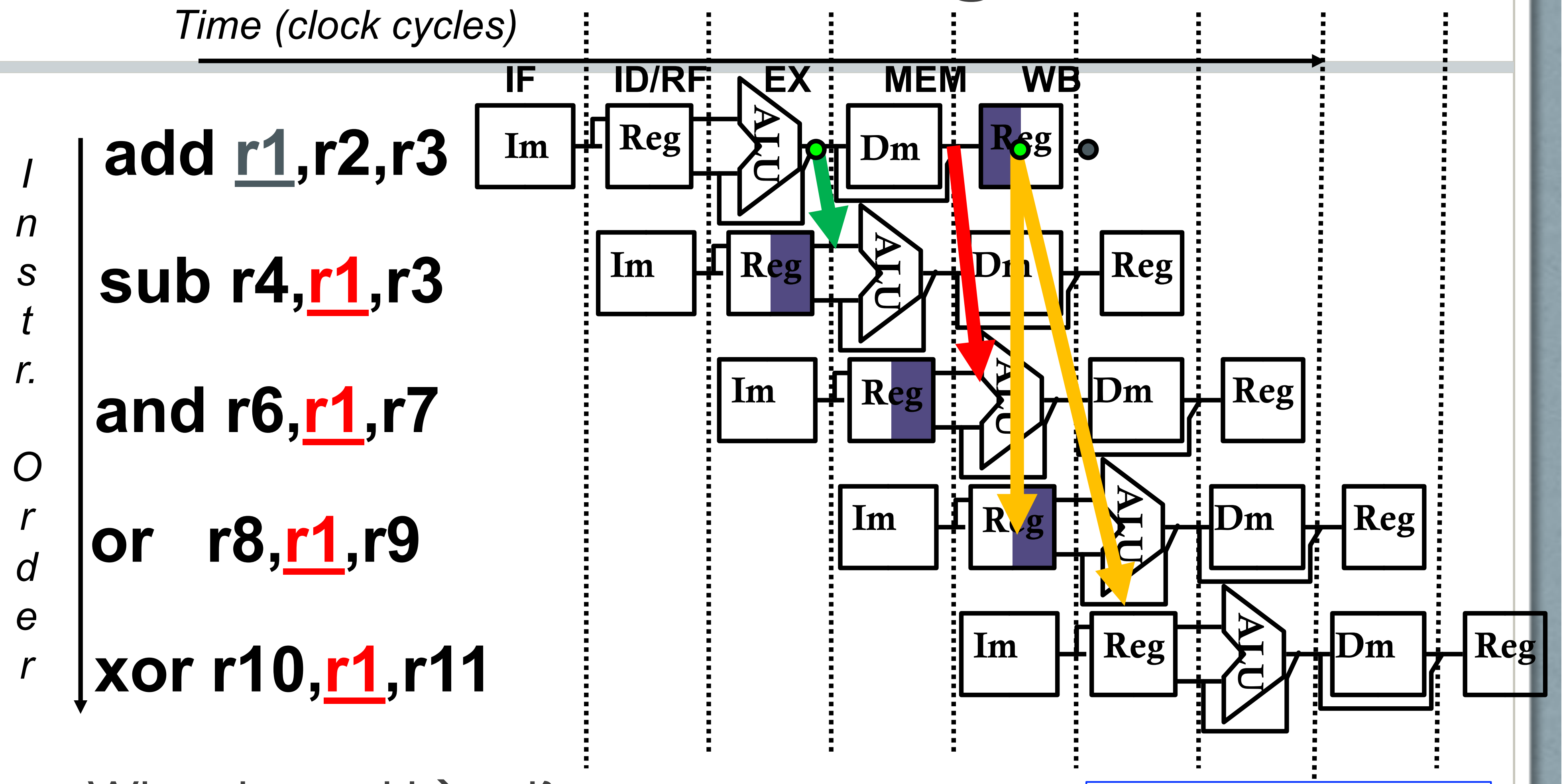


Forwarding



- Forwarding timing: EX out➔EX in (add→sub)
- What about add→and?

Forwarding



- What about add→and?
  - Pass it to MEM stage, then MEM out→EX in

Four data flow paths;
Two forwardings

Forwarding Examples

- Forwarding from EX/MEM

```
sub $2, $1, $3   IF  ID  EX  MEM  WB
and $12, $2, $5      IF  ID  EX   MEM  WB
```

- Forwarding from MEM/WB

```
sub $2, $1, $3   IF  ID  EX  MEM  WB
sub $5, $5, $6       IF  ID  EX   MEM  WB
and $12, $2, $5          IF  ID   EX   MEM  WB
```

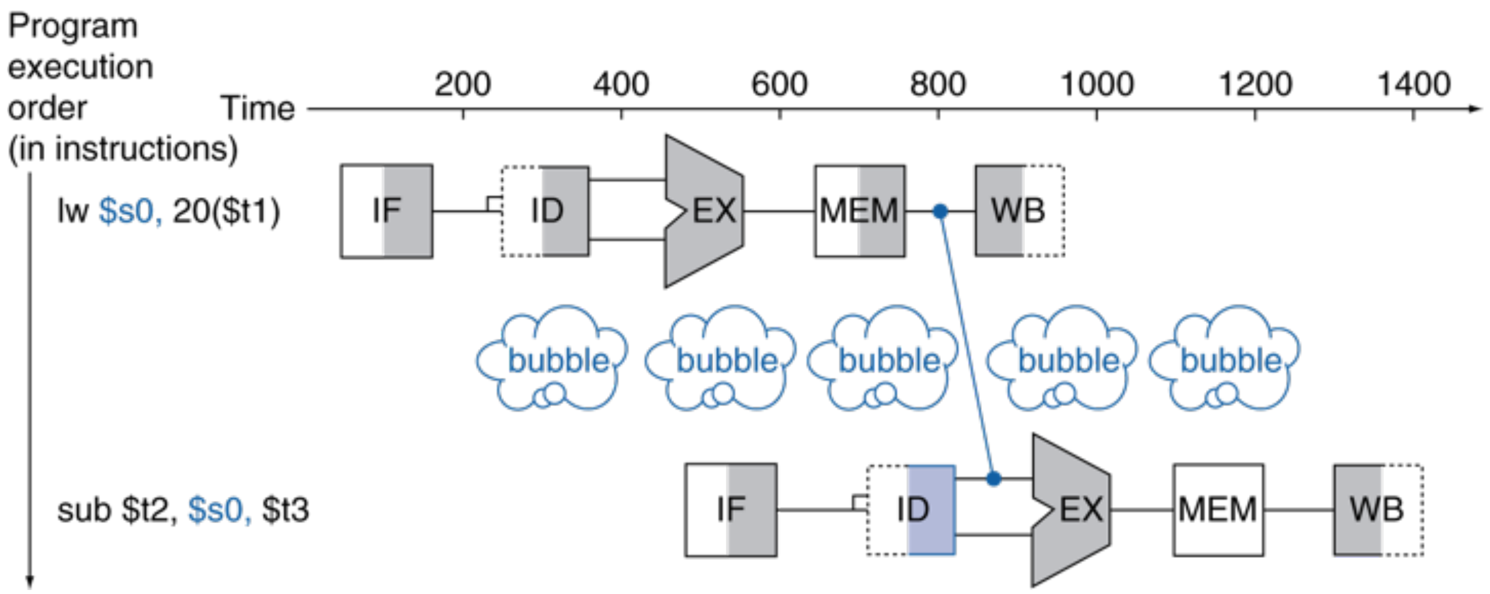
- Forwarding from ?

```
sub $2, $1, $3   IF  ID  EX  MEM  WB
sub $2, $2, $6       IF  ID  EX   MEM  WB
and $12, $2, $5          IF  ID   EX   MEM  WB
```

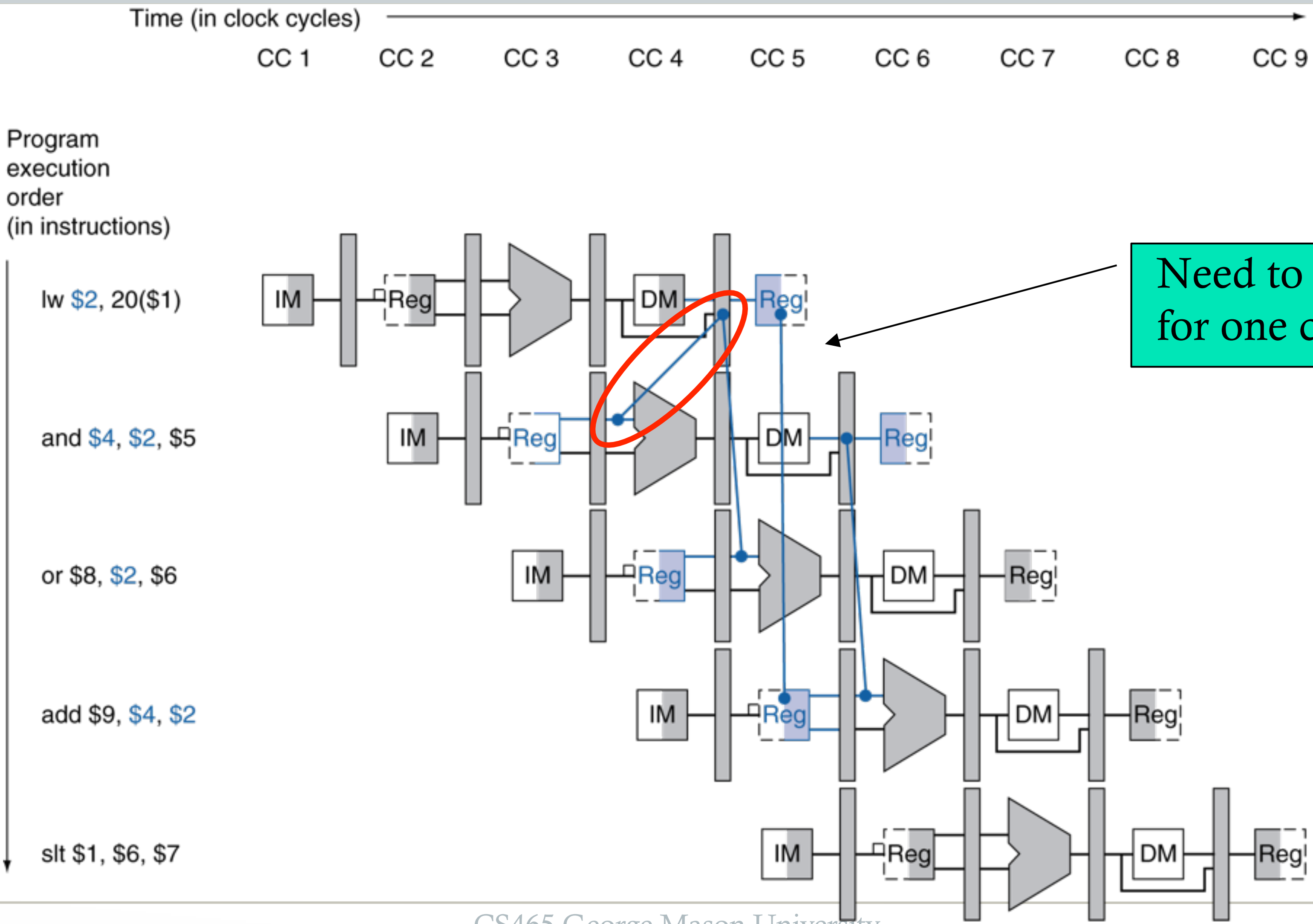
Double forwarding: give priority to the later producer

Load-Use Data Hazard

- Can’t always avoid stalls by forwarding
  - If value not computed when needed
  - Can’t forward backward in time!



Load-Use Data Hazard

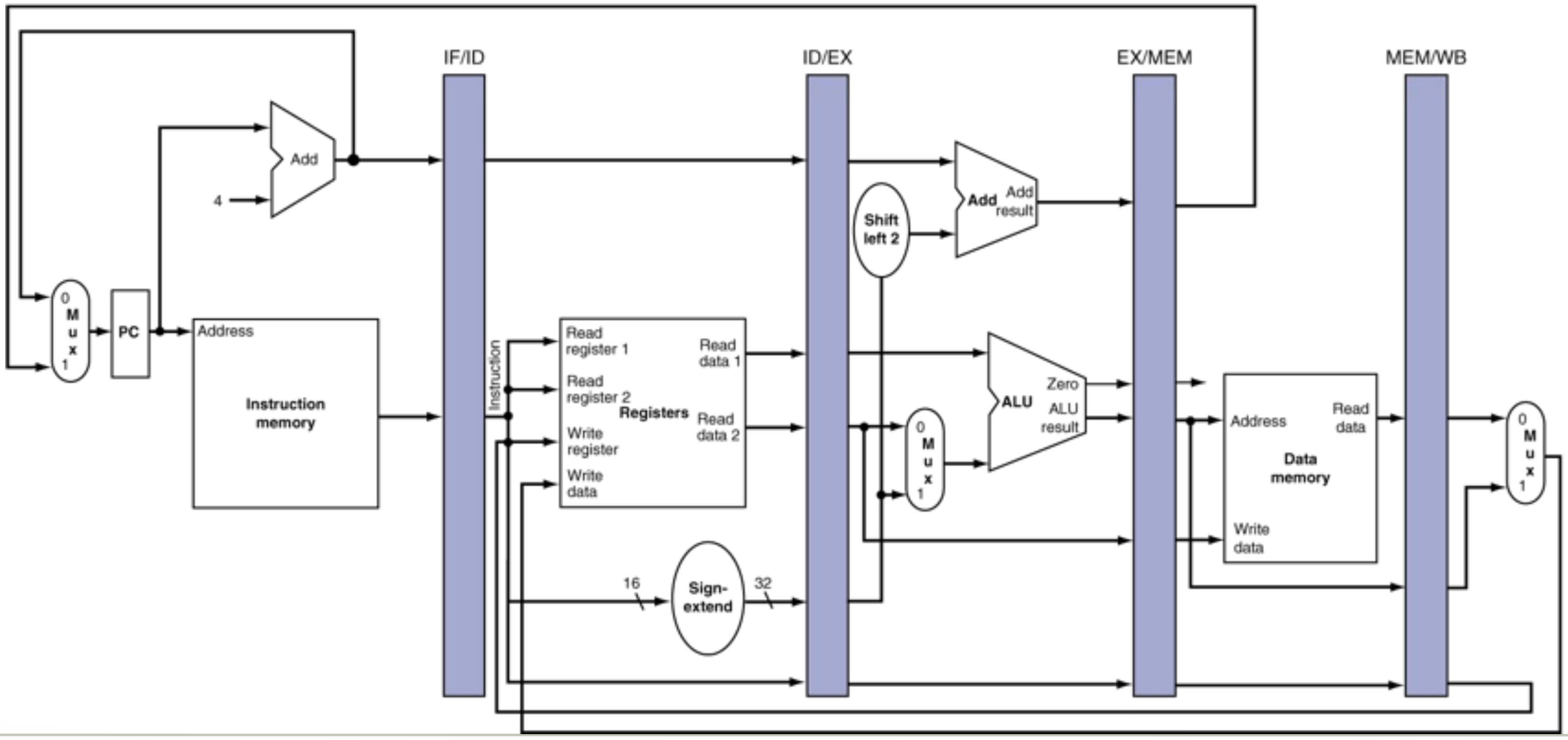


Need to stall for one cycle.

How to Stall Pipeline

Cycle N:

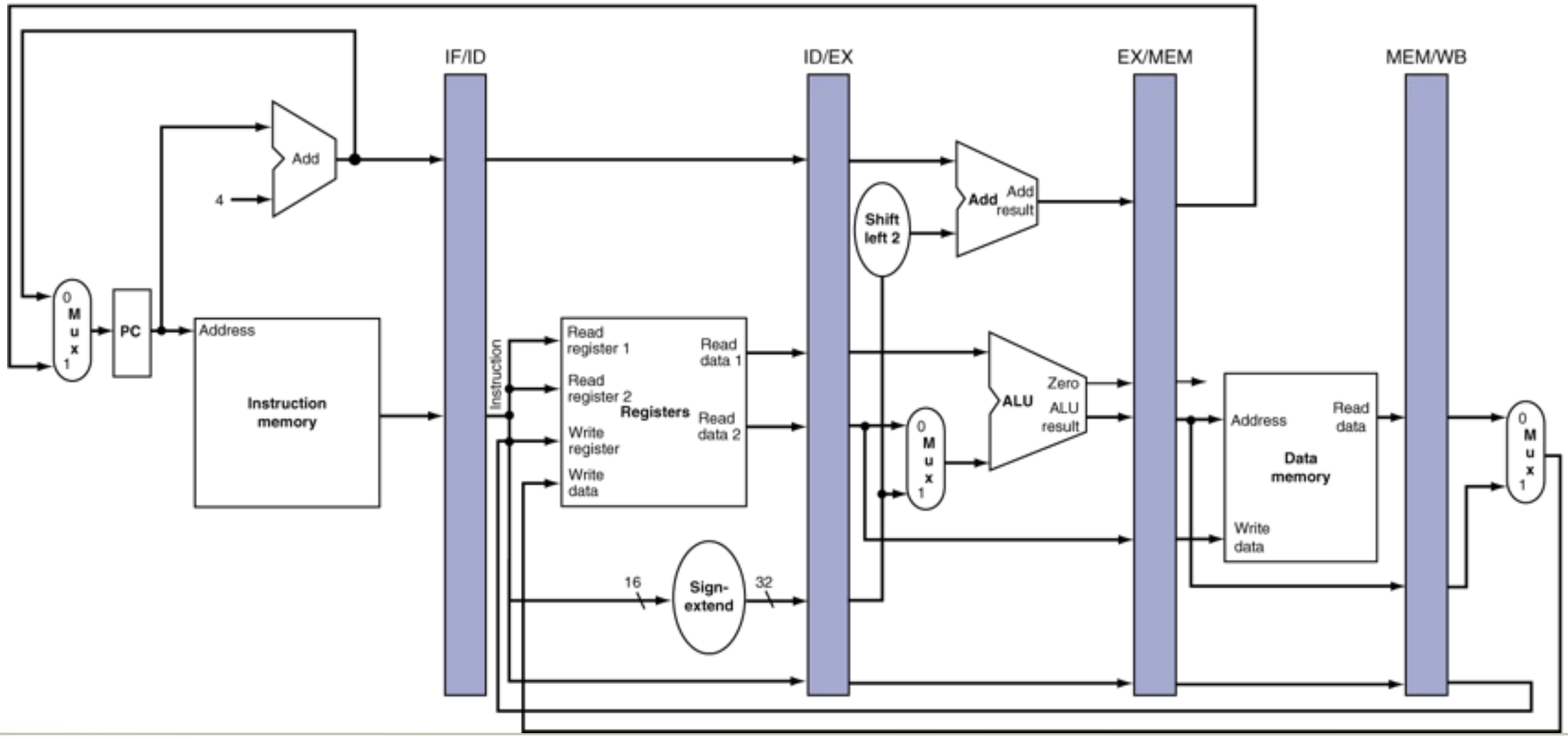
or … and $4, $2, $5 lw $2, 20($1)



How to Stall Pipeline

Cycle N+1: stay (repeat) become idle move forward as normal

or … and $4, $2, $5 nop lw $2, 20($1)



How to Stall Pipeline

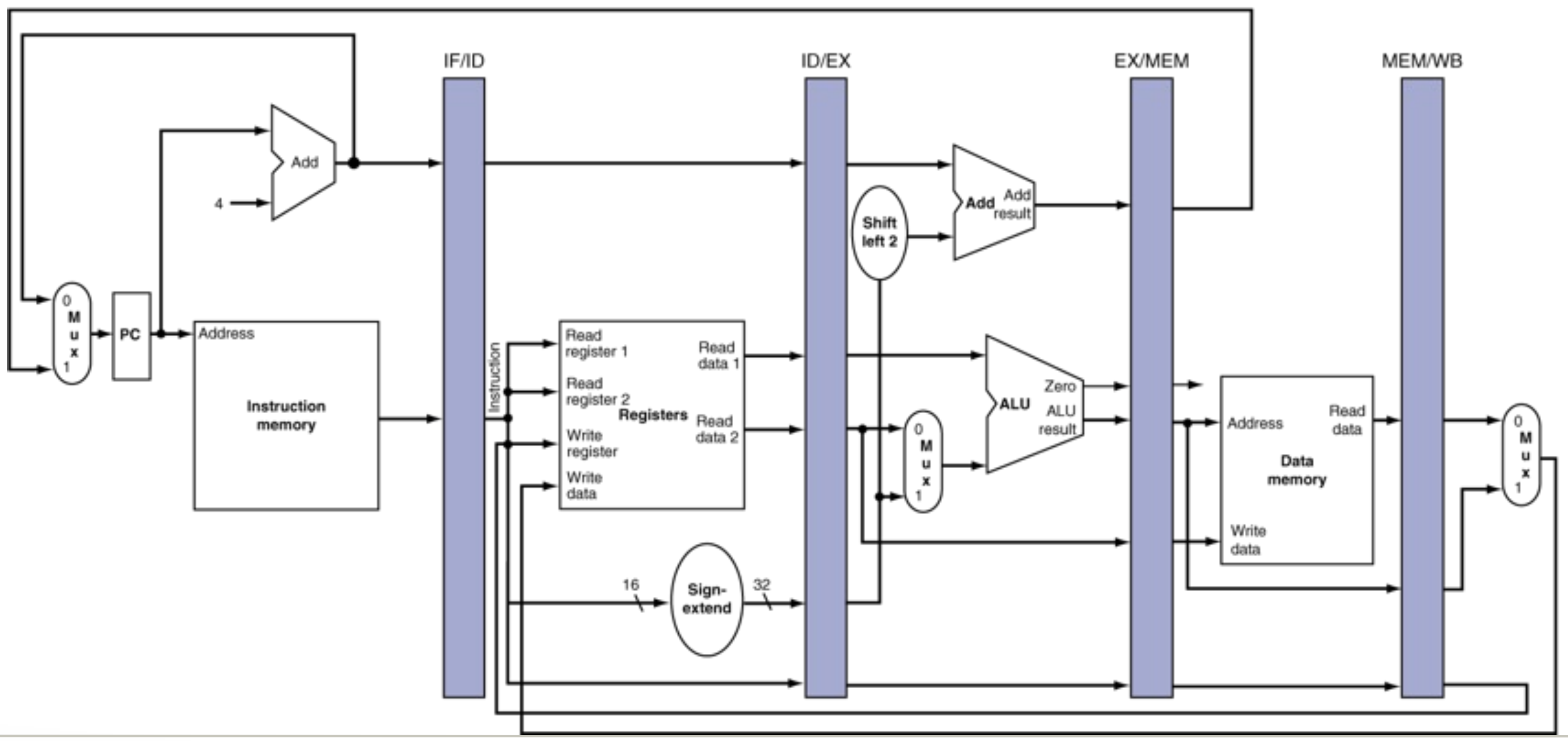
- Force all control values in ID/EX register to 0
  - Perform an `nop` (no-operation) for following stages
- Prevent update of PC and IF/ID register
  - Keep IF/ID unchanged – repeat the previous cycle
  - Keep PC unchanged – refetch the same instruction (the following instruction)
  - Add PCWrite and IF/IDWrite control to data hazard detection logic
  - 1-cycle stall allows MEM to read data for `lw`
    - Can subsequently forward to EX stage

How to Stall Pipeline

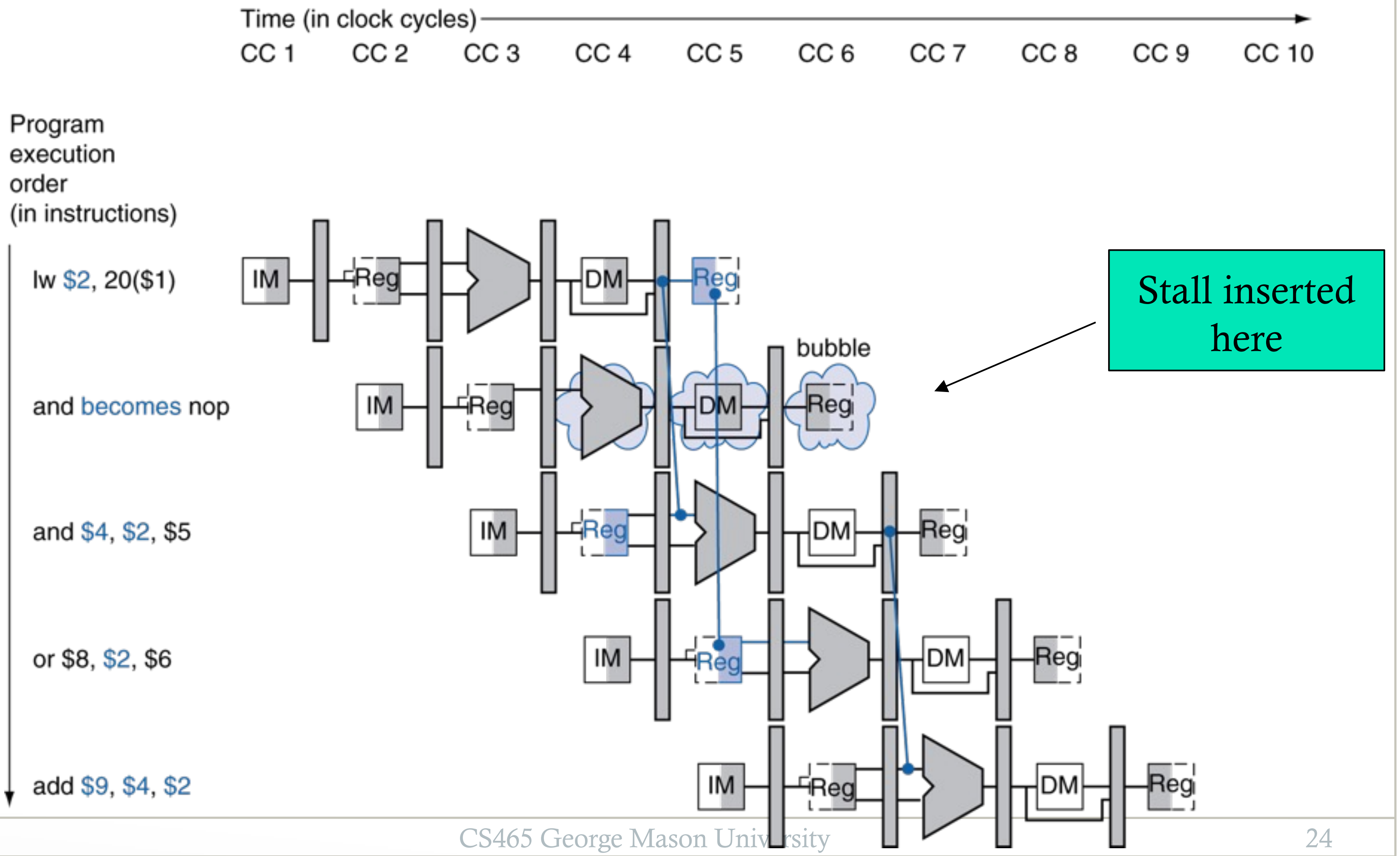
Cycle N+2: all move forward as normal

accept forwarding

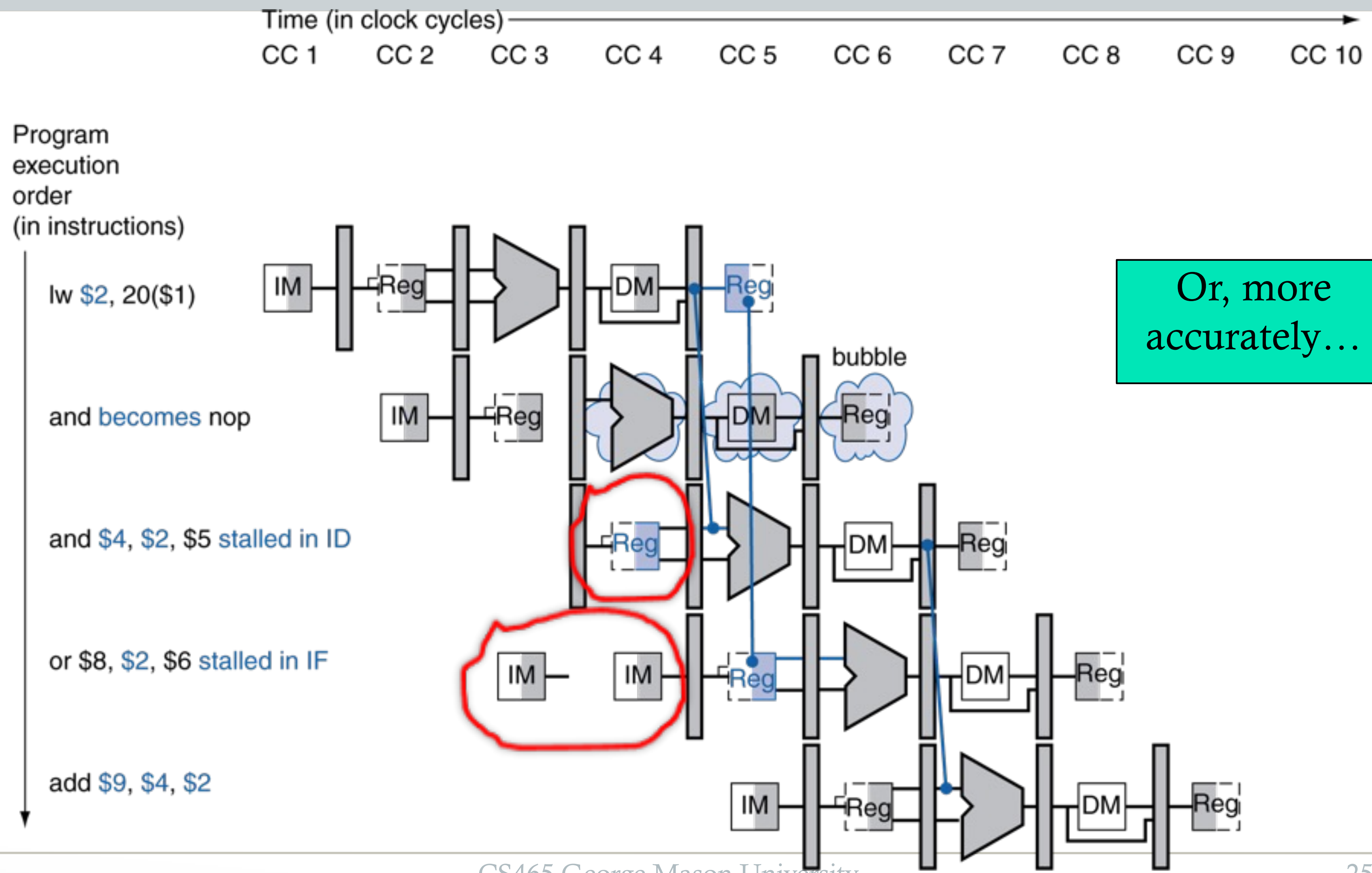
or … and $4, $2, $5 nop lw $2, 20($1)



Stall/Bubble in Pipeline

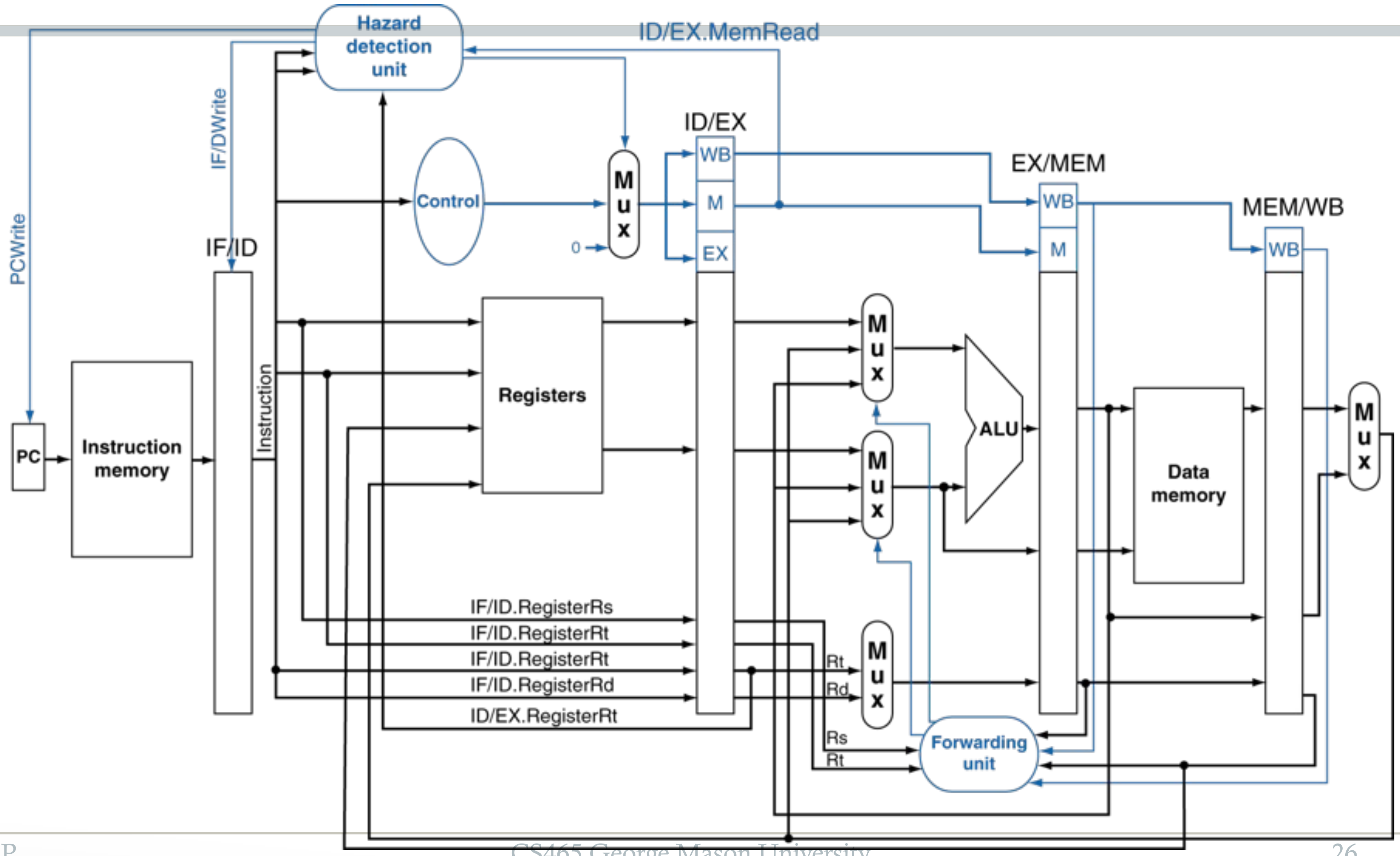


Stall/Bubble in Pipeline



Or, more accurately…

Advanced: Hazard Detection Unit


Hazard detection unit
ID/EX.MemRead
PCWrite
IF/DWrite
IF/ID
ID/EX
EX/MEM
MEM/WB
Control
0
M u x
WB
M
EX
PC
Instruction memory
Instruction
Registers
ALU
Data memory
IF/ID.RegisterRs
IF/ID.RegisterRt
IF/ID.RegisterRt
IF/ID.RegisterRd
ID/EX.RegisterRt
Rt
Rd
Rs
Rt
Forwarding unit