Facultad de Informática
Universidad Complutense de Madrid

INTRODUCTION TO COMPUTERS II

PIPELINED MICROARCHITECTURE: HAZARDS

Summary of hazards

- No structural hazards.
- There is a **data hazard** when executing an instruction with a source register that is the destination register of any of the 3 previous instructions.

	1	2	3	4	5	6	7	8
`addi t1,t2,5`	IF	ID	EX	M	**WB**			
`add t3,t1,t2`		IF	**ID**	EX	M	WB		
`addi t4,t1,15`			IF	**ID**	EX	M	WB	
`add t5,t1,t1`				IF	**ID**	EX	M	WB

- There is a **control hazard** when executing a **`jal`** or **`beq`** instruction.

	1	2	3	4	5	6
`beq t1,t2,L1`	IF	ID	**EX**	M	WB	
`???`		**IF**	ID	EX	M	WB

Pipelined processor without hazard management

(RF write at the <u>end</u> of the cycle)

- All hazards must be solved by software.
- **Data hazards**: depending on the case, insert 1, 2 or 3 **`nop`** instructions between the instruction that writes the register and the one that reads it.

	1	2	3	4	5	6	7	8	9
`addi t1,t2,5`	IF	ID	EX	M	**WB**				
`nop`		IF	ID	EX	M	WB			
`nop`			IF	ID	EX	M	WB		
`nop`				IF	ID	EX	M	WB	
`add t3,t1,t2`					IF	**ID**	EX	M	WB

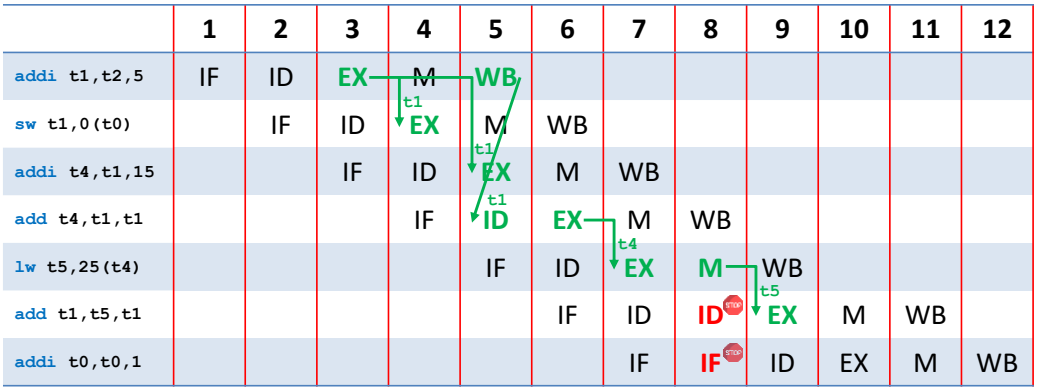
- **Control hazards**: insert 2 **`nop`** instructions after each branch instruction.

	1	2	3	4	5	6	7	8
`beq t1,t2,L1`	IF	ID	**EX**	M	WB			
`nop`		IF	ID	EX	M	WB		
`nop`			IF	ID	EX	M	WB	
`L1:addi t3,t1,1`				**IF**	ID	EX	M	WB

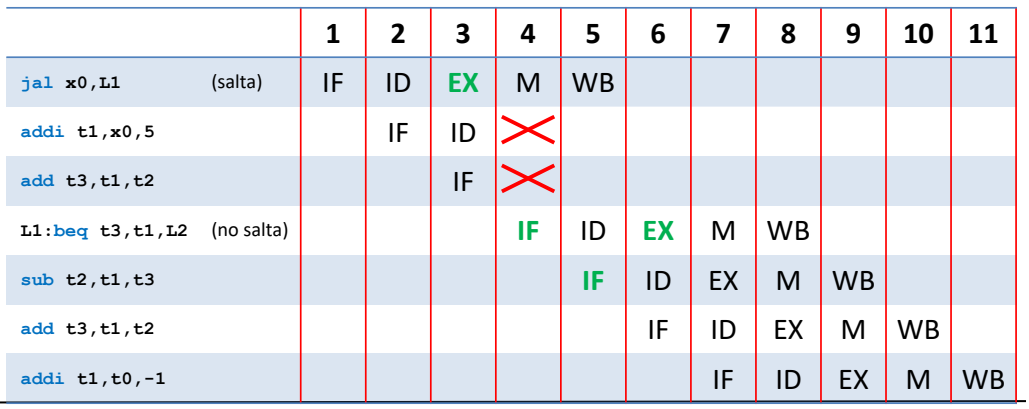
Pipelined processor with hazard management

(RF write in the middle of the cycle, forwarding unit and hazard unit with not-taken branch prediction)

- All hazards are solved by hardware.
- Rules to solve **data hazards**:
  1. If the instruction in EX has a source register that is the destination register of the instruction in MEM: the data is forwarded from MEM to EX.
  2. If the instruction in EX has a source register that is the destination register of the instruction in WB: the data is forwarded from WB to EX.
  3. If the instruction in EX has a source register that is the destination register of both the instructions in MEM and WB: the data is forwarded from MEM to EX.
  4. If the instruction in ID has a source register that is the destination register of the instruction in WB: the data is read from the RF since it has been written in the middle of the cycle.
  5. If the instruction is **`lw`** and has a destination register that is the source register of the instruction in ID: the instructions in IF and ID are stalled (the others advance as usual). There is a penalty of a one-cycle delay.



- Rules to solve **control hazards**:
  1. If the instruction in EX is **`jal`** or **`beq`** and the branch is taken: the instructions in IF and ID are flushed. There is a penalty of a two-cycle delay.
  2. If the instruction in EX is **`beq`** and the branch is not taken: the instructions advance as usual.



OTHER ALTERNATIVE PIPELINED MICROARCHITECTURES

Pipelined processor with optimized forwarding

(specific management of **`lw`→`sw`***)*

- The following rules to solve **data hazards** are added to the previous ones:
  6. If the instruction in EX is **`lw`** and has a destination register that is also the source register of a **`sw`** instruction in ID: no instruction is stalled.
  7. If the instruction in MEM is **`sw`** and has a source register that is also the destination register of a **`lw`** instruction in WB: the data is forwarded from WB to MEM.

	1	2	3	4	5	6
`lw s3,16(s0)`	IF	ID	EX	M	WB	
`sw s3,20(s0)`		IF	ID	EX	M (s3)	WB

Pipelined processor with hazard management through pipeline stalling

(RF write in the middle of the cycle and hazard unit through pipeline stalling)

- All hazards are solved by hardware.
- Rules to solve **data hazards**:
  1. If the instruction in ID has a source register that is also the destination register of the instructions in EX or MEM: the instructions in IF and ID are stalled (the others advance as usual). Depending on the case, there is a penalty of 1- or 2-cycle delay.
  2. If the instruction in ID has a source register that is also the destination register of the instruction in WB: the data is read from the RF since it has been written in the middle of the cycle.
- Rules to solve **control hazards**:
  1. If the instruction in ID or EX is **`jal`** or **`beq`**: the instruction in IF is stalled (the others advance as usual). There is a penalty of a two-cycle delay.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
`addi t1,t2,5`	IF	ID	EX	M	WB (t1)									
`add t3,t1,t2`		IF	ID (STOP)	ID (STOP)	ID	EX	M	WB						
`addi t4,t1,15`			IF (STOP)	IF (STOP)	IF	ID	EX	M	WB					
`beq t1,t2,L2`						IF	ID	EX	M	WB				
`lw t5,25(t4)`							IF (STOP)	IF (STOP)	IF	ID	EX	M	WB	
`add t1,t5,t1`										IF	ID	EX	M	WB

Pipelined processor without hazard management

(RF write in the middle of the cycle)

- All hazards are solved by software.
- **Data hazards**: depending on the case, insert 1 or 2 `nop` instructions between the instruction that writes the register and the one that reads it.

	1	2	3	4	5	6	7	8
`addi t1,t2,5`	IF	ID	EX	M	WB			
`nop`		IF	ID	EX	M	WB		
`nop`			IF	ID	EX	M	WB	
`add t3,t1,t2`				IF	ID (t1)	EX	M	WB

- **Control hazards**: insert 2 `nop` instructions after each branch instruction.

	1	2	3	4	5	6	7	8
`beq t1,t2,L1`	IF	ID	EX	M	WB			
`nop`		IF	ID	EX	M	WB		
`nop`			IF	ID	EX	M	WB	
`L1:addi t3,t1,1`				IF	ID	EX	M	WB