Pipeline II

Hazards

Pipeline with 5 stages

- **Fetch instruction (FI):** Read the next expected instruction into a buffer.
- **Decode instruction (DI):** Determine the opcode and the operand specifiers.
- **Fetch operands (FO):** Fetch each operand.
- **Execute instruction (EI):** Perform the indicated operation.
- **Write operand (WO):** Store the result in register or memory.

Pipeline with 5 stages

	1st	2nd	3rd	4th	5th	6th	7th
add ax, bx	FI	DI	FO	EI	WO		
add cx, dx		FI	DI	FO	EI	WO	
sub si, di			FI	DI	FO	EI	WO

Pipeline Hazard in Computer

- Problems with the instruction pipeline in CPU when the next instruction cannot execute in the following clock cycle and can potentially lead to incorrect computation results.

- Three Hazard types:
  - Structural/Resource
    - same resource is needed multiple times in the same cycle
  - Data
    - Instruction depends on result of prior instruction still in the pipeline data dependencies limit pipelining
  - Control
    - Next executed instruction may not be the next specified instruction

Structural/Resource Hazard

- A resource hazard occurs when two (or more) instructions that are already in the pipeline need the same resource.
- The result is that the instructions must be executed in serial rather than parallel for a portion of the pipeline.
- Examples:
  - Two accesses to a single ported memory
- Solutions:
  - Increase available resources
  - Stalling

Structural/Resource Hazard

Assume we have a Single Port Memory

add ax, [num1]	FI	DI	FO	EI	WO		
add cx, dx		FI	DI	FO	EI	WO	
sub ax, dx			FI	DI	FO	EI	WO

Structural/Resource Hazard

Solution

Stall - Waste one clock cycle

add ax, [num1]	FI	DI	FO	EI	WO			
add cx, dx		FI	DI	FO	EI	WO		
sub ax, dx			Stall	FI	DI	FO	EI	WO

Data Hazards

- Data hazards occur when instructions that exhibit data dependence modify data in different stages of a pipeline.

- Three types of data hazards
  - RAW: Read after Write or Flow dependency
  - WAR: Write after Read or anti-dependency
  - WAW: Write after Write or Output dependency

Data Hazards

- RAW: Read after Write or Flow dependency
  - add ax, cx
  - add bx, ax
- WAW: Write after Write or output dependency
  - add ax, bx
  - add ax, cx
- WAR: Write after Read or anti-dependency
  - add ax, bx
  - add bx, cx

Read after Write (RAW) or flow dependency

Next Instruction tries to read operand before first Instruction writes it

add ax, bx	FI	DI	FO	EI	WO		
add cx, ax		FI	DI	FO	EI	WO	

Read after Write (RAW) or flow dependency

- Solution 1
  Stalling – waste two clock cycles and fetch operand after WO stage of 1st instruction

add **ax**, bx	FI	DI	FO	EI	WO			
add cx, **ax**		FI	DI	Stall	Stall	FO	EI	WO

Read after Write (RAW) or flow dependency

- Solution 2

  Data Forwarding
  - Used in RAW dependency, data is forwarded to next instruction as soon as its available so that next instruction doesn't have to wait for write operand
  - Needs changes in hardware
  - Needs only one stall

add ax, bx	FI	DI	FO	EI	WO		
add cx, ax		FI	DI	Stall	FO	EI	WO

Write after write (WAW), or output dependency

- Two instructions both write to the same location.

- A hazard occurs if the write operations take place in the reverse order of the intended sequence.
    - I: write to location X
    - J: write to location X
  - Incorrect answer if J finishes before I

- Can not occur in this pipeline

Write after read (WAR) or antidependency

- An instruction reads a register or memory location and a succeeding instruction writes to the location.
- A hazard occurs if the write operation completes before the read operation takes place.
    - I: read from location X
    - J: write to location X
  - Incorrect answer if J finishes before I

- Can not occur in this pipeline

Control Hazard

- A control hazard, also known as a branch hazard
- Occurs when the pipeline makes the wrong decision on a branch prediction and therefore brings instructions into the pipeline that must subsequently be discarded

Control Hazard

Instruction 1

Instruction 2 (Conditional Branch Instruction to label1)

Instruction 3

Instruction 4

Instruction 5

label1: Instruction 6

Assuming no RAW or structural hazard.

	1
Instruction 1	FI
Branch Instruction	
Instruction 3	
Instruction 4	
Instruction 5	
label1: Instruction 6	

	1	2
Instruction 1	FI	DI
Branch Instruction		FI
Instruction 3		
Instruction 4		
Instruction 5		
label1: Instruction 6		

- Instruction 2 is conditional branching

	1	2	3
Instruction 1	FI	DI	FO
Branch Instruction		FI	DI
Instruction 3			?
Instruction 4			
Instruction 5			
label1: Instruction 6			

- Until the instruction is actually executed, it is impossible to determine whether the branch will be taken or not.

Dealing with Branches

- How to handle Branches? Solution: Branch Prediction
- **Branch Prediction Strategies:**

1. **Always Untaken** i.e. fetch next instruction
2. **Always Taken** i.e. fetch instruction whose offset is in the instruction
3. **Delayed Branch** i.e. add some code between branch instruction and target instruction
4. **Dynamic Prediction** on the basis of history of a branch

Dealing with Branches

- How to handle Branches? Solution: Branch Prediction
- **Branch Prediction Strategies:**

1. **Always Untaken** i.e. fetch next instruction
2. **Always Taken** i.e. fetch instruction whose offset is in the instruction
3. **Delayed Branch** i.e. add some code between branch instruction and target instruction
4. **Dynamic Prediction** on the basis of history of a branch

	1	2	3
Instruction 1	FI	DI	FO
Branch Instruction		FI	DI
Instruction 3			FI
Instruction 4			
Instruction 5			
label1: Instruction 6			

- Processor predicts that the branch is not taken and fetch next instruction.

	1	2	3	4	5
Instruction 1	FI	DI	FO	EI	WO
Branch Instruction		FI	DI	FO	EI
Instruction 3			FI	DI	FO
Instruction 4				FI	DI
Instruction 5					FI
label1: Instruction 6					

- What if branch prediction is wrong?
- All work from Instruction 3-5 is wasted.

	1	2	3	4	5	6	7	8	9	10
Instruction 1	FI	DI	FO	EI	WO					
Branch Instruction		FI	DI	FO	EI	WO				
Instruction 3			FI	DI	FO					
Instruction 4				FI	DI					
Instruction 5					FI					
label1: Instruction 6						FI	DI	FO	EI	WO

- How many cycles are wasted?

Control Hazard

- A control hazard, also known as a branch hazard

- Occurs when the pipeline makes the wrong decision on a branch prediction and therefore brings instructions into the pipeline that must subsequently be discarded

Summary: Pipelining Hazards

- Resource/Structural Hazard
  - Add more resource
  - Stall

- Data Hazards
  - Stall
  - Data Forwarding

- Control Hazard
  - Stall