Pipelining

Last modified 8/26/24

Pipelining

- Basic Concept
  - Break complex tasks into a series of simpler serial tasks
  - As each simple task completes – move to the next task
  - In parallel, start the current task with new material
  - Laundry example
    - Complex task – do your laundry
    - Simple tasks – wash(30min), dry(30 min), fold(10min)
    - 3 loads, 1 washer, 1 dryer

wash1 – dry1 – fold1	wash2 – dry2 – fold2	wash3 – dry3 – fold3	
70min	70min	70min	210min

wash1	dry1	fold1			
	wash2	dry2	fold2		
		wash3	dry3	fold3	
30min	30min	30min	30min	10min	130min

Pipelining

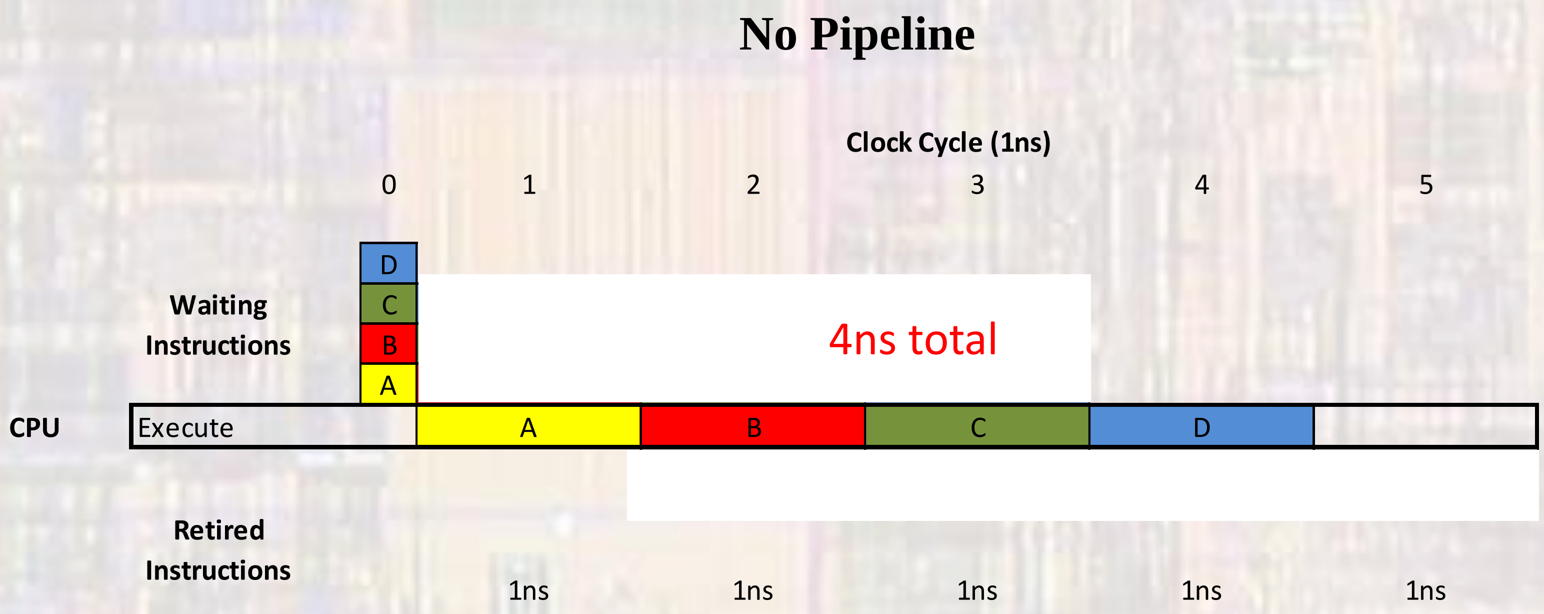
- Pipeline our single cycle processor

  - 5 Stages of Instruction Execution
    - Fetch (IF)
    - Decode / Register Access (ID)
    - Execute (EX)
    - Memory Access (MEM)
    - Write Back (WB)

Pipeline these at 1 stage each

Pipeling

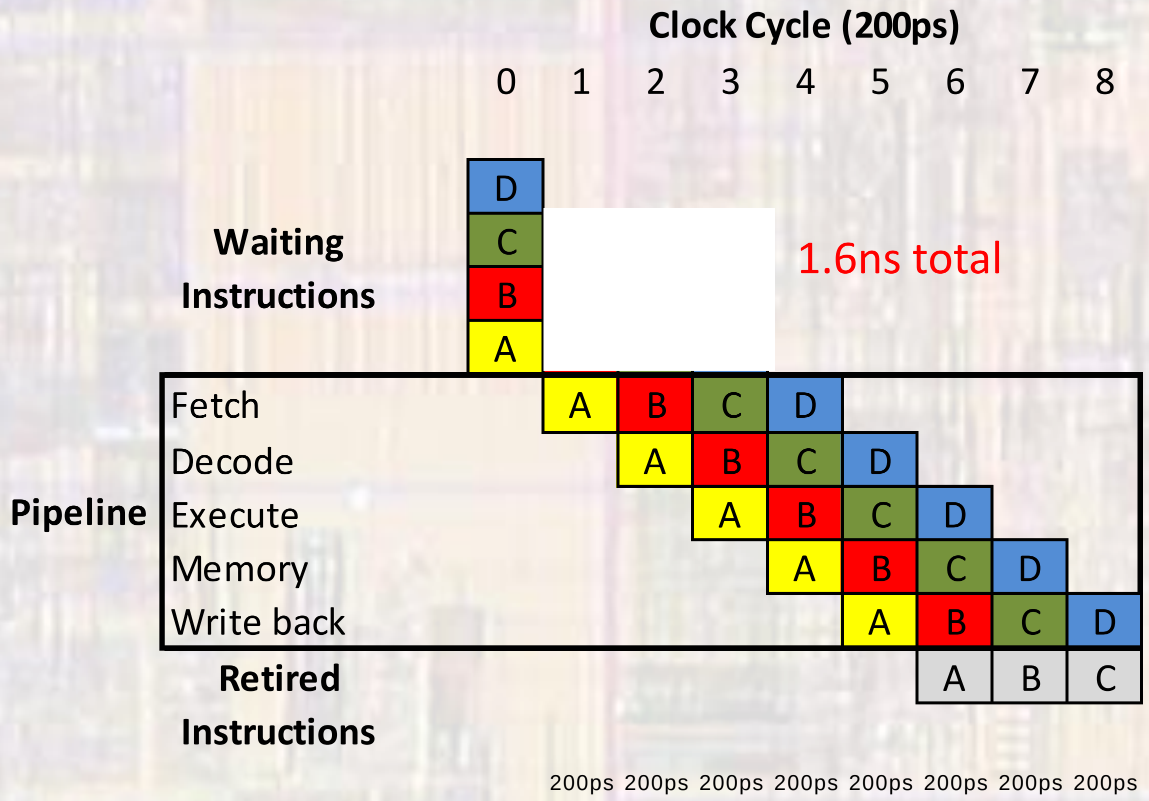
- No Pipeline
  - Complete each instruction before starting the next
  - 1ns to complete each instruction



Execute = fetch instruction, decode, execute, mem, write back

Pipelining

- Pipelining
  - Break complex tasks into smaller chunks
  - Start the next instruction as soon as each subtask is complete



Pipelining

- Pipeline Performance
  - Pipelining does not reduce the time to execute an instruction (1ns in this example)
    - In fact – it usually increases the instruction execution time due to costs of implementing the pipeline
  - Pipelining does increase the instruction throughput
    - 1 instruction completes every 200ps

No Pipeline

Time	1000	1000	1000
IF/ID/EX/MEM/WB	1	2	3

1 inst

Pipeline

Time	200	200	200	200	200	200	200	200	200	200	200	200	200	200	200
IF	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ID		1	2	3	4	5	6	7	8	9	10	11	12	13	14
EX			1	2	3	4	5	6	7	8	9	10	11	12	13
MEM				1	2	3	4	5	6	7	8	9	10	11	12
WB					1	2	3	4	5	6	7	8	9	10	11

1 inst

Pipelining

- Pipeline Performance

5 stages
1000ps non-pipelined
200ps/stage pipelined

  - Non-pipelined
    - 1M Instructions → 1M * 1000ps = 1ms
  - Pipelined (5 stage)
    - 1M Instructions → Fill Time + Execute time
    - 1M Instructions → 4 * 200ps + 1M * 200ps ≅ 200us
  - Faster completion time: 1/5
  - Overall throughput improvement of 5x

Pipelining

- Pipeline Performance – with penalty

5 stages
1000ps non-pipelined
240ps/stage pipelined

- Non-pipelined
  - 1M Instructions → 1M * 1000ps = 1ms

- Pipelined (5 stage w/20% penalty per stage)
  - 1M Instructions → Fill Time + Execute time
  - 1M Instructions → 4 * 240ps + 1M * 240ps ≅ 240us

- Faster completion time: 1/4.2
- Overall throughput improvement of 4.2x

Pipelining

- Pipeline Performance

  - Not all instructions need to use all the processing stages

Instruction	IF	ID/RR	EX	MEM	WB
ADD	X	X	X		X
OR	X	X	X		X
LW	X	X	X	X	X
SW	X	X	X	X	
BEQ	X	X	X		

3, 4, or 5 stages required

  - Can’t take advantage of this in either case because we need a consistent clock frequency

Pipelining

- Pipeline Performance

  - Processing stages typically do not all take the same amount of time

Stage	IF	ID/RR	EX	MEM	WB
Delay	200ps	100ps	200ps	200ps	100ps

  - Non-pipelined
    - 800ps clock period
  - Pipelined
    - Need to account for worst case cycle time
    - 200ps clock period

Pipelining

- Pipeline Performance

  - Non-pipelined
    - 1M Instructions → 1M * 800ps = 800us

  - Pipelined (5 stage w/20% penalty per stage)
    - 1M Instructions → Fill Time + Execute time
    - 1M Instructions → 4 * 240ps + 1M * 240ps ≅ 240us

  - Faster completion time: 240/800
  - Overall throughput improvement of 3.33x

Pipelining

- Pipeline Performance

15 stages
800ps non-pipelined
(1000ps/15)*1.2 = 84ps/stage pipelined

  - Non-pipelined
    - 1M Instructions → 1M * 800ps = 800us
  - Pipelined (15 stage w/20% penalty per stage)
    - 1M Instructions → Fill Time + Execute time
    - 1M Instructions → 14 * 84ps + 1M * 84ps ≅ 84us
  - Faster completion time: 84/800
  - Overall throughput improvement of 9.52x