INCHEON NATIONAL UNIVERSITY

IAA6007: Computer Architecture
Ch.9. Pipeline and Vector Processing

Wooil Kim
Dept. of Computer Science & Engineering
Incheon National University

2019 Fall

Outline

- 9.1 Parallel processing
- 9.2 Pipelining
- 9.4 Instruction pipeline
- 9.5 RISC pipeline

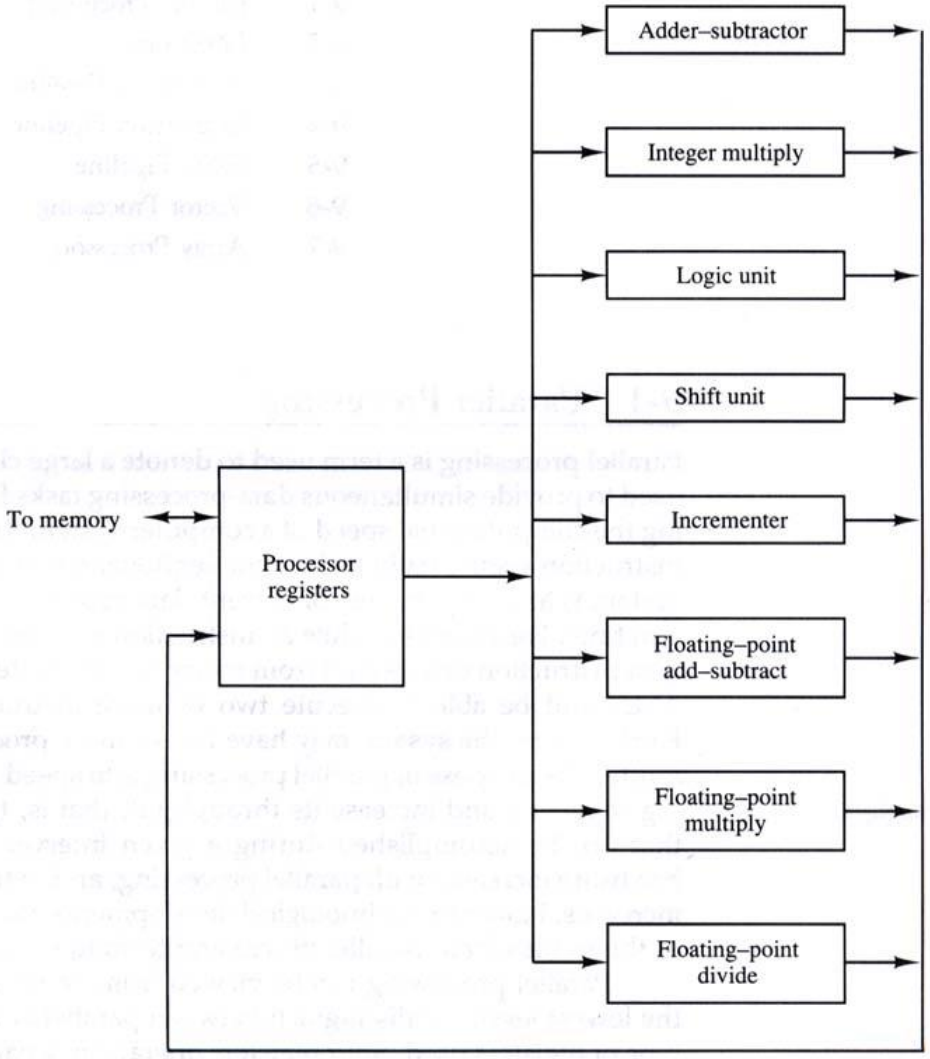
Parallel processing

- Parallel processing
  - A large class of techniques that are used to provide simultaneous data processing task
    - Bit level parallelism – serial/parallel operations by register (shift/parallel load)
    - Instruction level parallelism – functional unit level
- Flynn's taxonomy (= classification, 1996)
  - Instruction stream - instruction sequence read from memory
  - Data stream – operations performed on the data
  - Single Instruction stream, Single Data stream (SISD)
    - Single control, processor, & memory unit
  - Single Instruction stream, Multiple Data stream (SIMD)
    - Common control unit, shared memory unit
  - Multiple Instruction stream, Single Data stream (MISD)
    - Theoretical interest
  - Multiple Instruction stream, Multiple Data stream (MIMD)
    - Multiprocessor, multi-computer system

Parallel processing

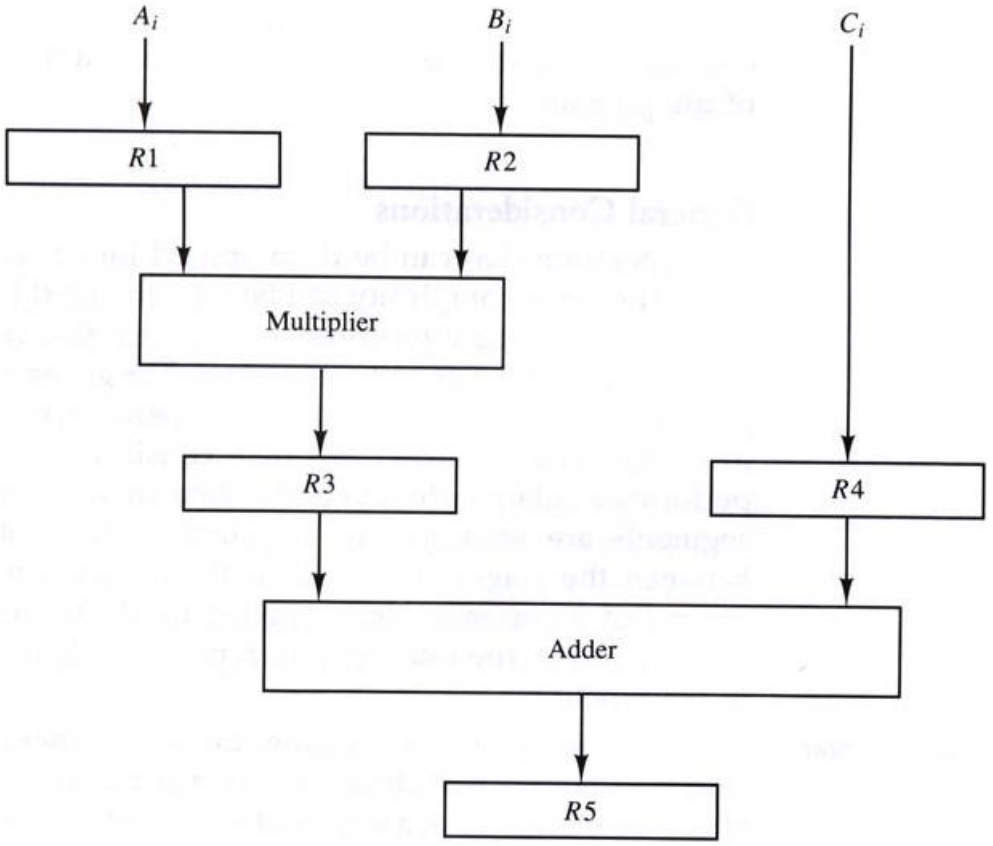
- Processor with multiple functional units

Figure 9-1 Processor with multiple functional units.



Pipelining

- Pipelining
  - Technique of decomposing a sequential process into sub-operations with each sub-operations being executed in a special dedicated segment
  - Ex) $A_i * B_i + C_i$ for i = 1, 2, 3, …, 7
    $R1 \leftarrow A_i$, $R2 \leftarrow B_i$
    $R3 \leftarrow R1 * R2$, $R4 \leftarrow C_i$
    $R5 \leftarrow R3 + R4$



Pipelining

- Example

TABLE 9-1 Content of Registers in Pipeline Example

Clock Pulse Number	Segment 1		Segment 2		Segment 3
	R1	*R2*	*R3*	*R4*	*R5*
1	A_1	B_1	—	—	—

Pipelining

- Example

TABLE 9-1 Content of Registers in Pipeline Example

Clock Pulse Number	Segment 1		Segment 2		Segment 3
	R1	R2	R3	R4	R5
1	A_1	B_1	—	—	—
2	A_2	B_2	$A_1 * B_1$	C_1	—

Pipelining

- Example

TABLE 9-1 Content of Registers in Pipeline Example

Clock Pulse Number	Segment 1		Segment 2		Segment 3
	R1	R2	R3	R4	R5
1	A_1	B_1	—	—	—
2	A_2	B_2	$A_1 * B_1$	C_1	—
3	A_3	B_3	$A_2 * B_2$	C_2	$A_1 * B_1 + C_1$

Pipelining

- Example

TABLE 9-1 Content of Registers in Pipeline Example

Clock Pulse Number	Segment 1		Segment 2		Segment 3
	R1	*R2*	*R3*	*R4*	*R5*
1	A_1	B_1	—	—	—
2	A_2	B_2	$A_1 * B_1$	C_1	—
3	A_3	B_3	$A_2 * B_2$	C_2	$A_1 * B_1 + C_1$
4	A_4	B_4	$A_3 * B_3$	C_3	$A_2 * B_2 + C_2$

Pipelining

- Example

TABLE 9-1 Content of Registers in Pipeline Example

Clock Pulse Number	Segment 1		Segment 2		Segment 3
	R1	*R2*	*R3*	*R4*	*R5*
1	A_1	B_1	—	—	—
2	A_2	B_2	$A_1 * B_1$	C_1	—
3	A_3	B_3	$A_2 * B_2$	C_2	$A_1 * B_1 + C_1$
4	A_4	B_4	$A_3 * B_3$	C_3	$A_2 * B_2 + C_2$
5	A_5	B_5	$A_4 * B_4$	C_4	$A_3 * B_3 + C_3$

Pipelining

- Example

TABLE 9-1 Content of Registers in Pipeline Example

Clock Pulse Number	Segment 1		Segment 2		Segment 3
	R1	R2	R3	R4	R5
1	A_1	B_1	—	—	—
2	A_2	B_2	$A_1 * B_1$	C_1	—
3	A_3	B_3	$A_2 * B_2$	C_2	$A_1 * B_1 + C_1$
4	A_4	B_4	$A_3 * B_3$	C_3	$A_2 * B_2 + C_2$
5	A_5	B_5	$A_4 * B_4$	C_4	$A_3 * B_3 + C_3$
6	A_6	B_6	$A_5 * B_5$	C_5	$A_4 * B_4 + C_4$

Pipelining

- Example

TABLE 9-1 Content of Registers in Pipeline Example

Clock Pulse Number	Segment 1		Segment 2		Segment 3
	R1	*R2*	*R3*	*R4*	*R5*
1	A_1	B_1	—	—	—
2	A_2	B_2	$A_1 * B_1$	C_1	—
3	A_3	B_3	$A_2 * B_2$	C_2	$A_1 * B_1 + C_1$
4	A_4	B_4	$A_3 * B_3$	C_3	$A_2 * B_2 + C_2$
5	A_5	B_5	$A_4 * B_4$	C_4	$A_3 * B_3 + C_3$
6	A_6	B_6	$A_5 * B_5$	C_5	$A_4 * B_4 + C_4$
7	A_7	B_7	$A_6 * B_6$	C_6	$A_5 * B_5 + C_5$

Pipelining

- Example

TABLE 9-1 Content of Registers in Pipeline Example

Clock Pulse Number	Segment 1		Segment 2		Segment 3
	R1	R2	R3	R4	R5
1	A_1	B_1	—	—	—
2	A_2	B_2	$A_1 * B_1$	C_1	—
3	A_3	B_3	$A_2 * B_2$	C_2	$A_1 * B_1 + C_1$
4	A_4	B_4	$A_3 * B_3$	C_3	$A_2 * B_2 + C_2$
5	A_5	B_5	$A_4 * B_4$	C_4	$A_3 * B_3 + C_3$
6	A_6	B_6	$A_5 * B_5$	C_5	$A_4 * B_4 + C_4$
7	A_7	B_7	$A_6 * B_6$	C_6	$A_5 * B_5 + C_5$
8	—	—	$A_7 * B_7$	C_7	$A_6 * B_6 + C_6$

Pipelining

- Example

TABLE 9-1 Content of Registers in Pipeline Example

Clock Pulse Number	Segment 1		Segment 2		Segment 3
	$R1$	$R2$	$R3$	$R4$	$R5$
1	A_1	B_1	—	—	—
2	A_2	B_2	$A_1 * B_1$	C_1	—
3	A_3	B_3	$A_2 * B_2$	C_2	$A_1 * B_1 + C_1$
4	A_4	B_4	$A_3 * B_3$	C_3	$A_2 * B_2 + C_2$
5	A_5	B_5	$A_4 * B_4$	C_4	$A_3 * B_3 + C_3$
6	A_6	B_6	$A_5 * B_5$	C_5	$A_4 * B_4 + C_4$
7	A_7	B_7	$A_6 * B_6$	C_6	$A_5 * B_5 + C_5$
8	—	—	$A_7 * B_7$	C_7	$A_6 * B_6 + C_6$
9	—	—	—	—	$A_7 * B_7 + C_7$

Pipelining

- General considerations
  - Task – total operation performed going through all the segments in the pipeline

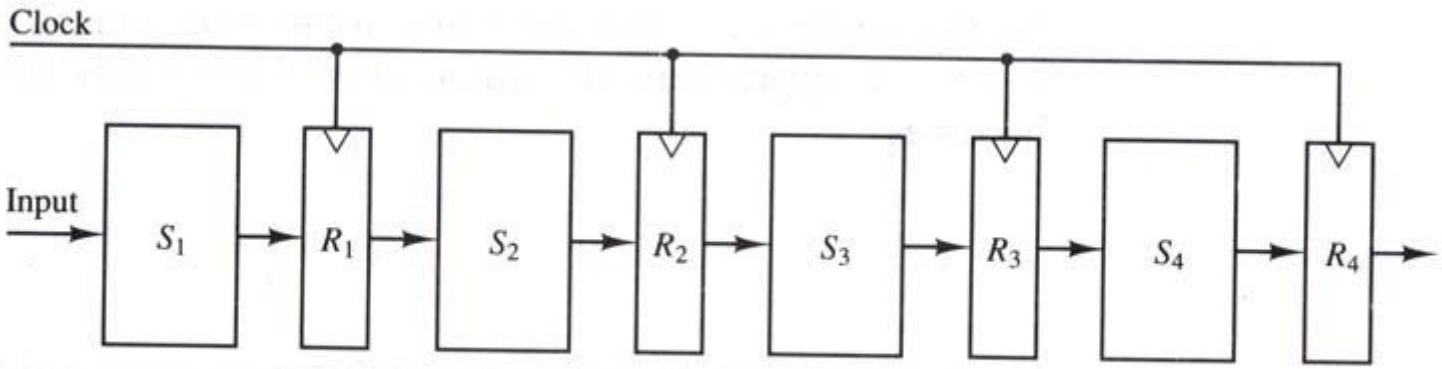
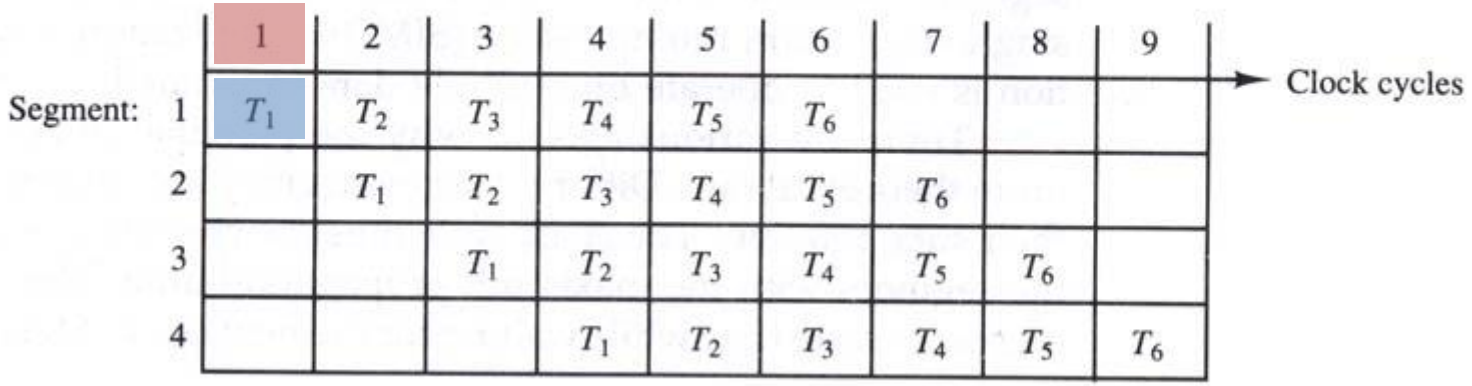


Figure 9-3 Four-segment pipeline.

Pipelining

- General considerations (cont.)
  - Space-time diagram

Figure 9-4 Space-time diagram for pipeline.

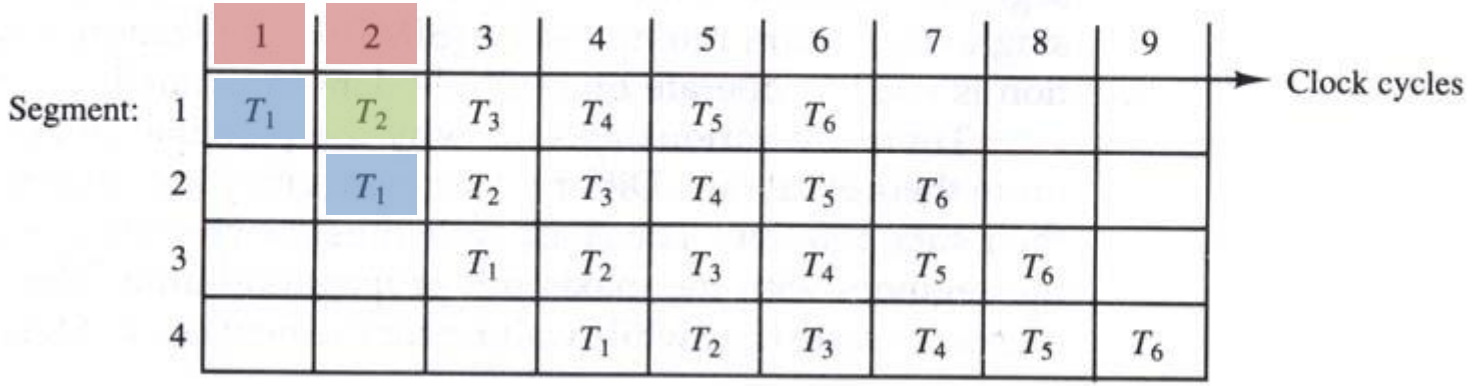


Six tasks T_1 to T_6 execute in four segments

Pipelining

- General considerations (cont.)
  - Space-time diagram

Figure 9-4 Space-time diagram for pipeline.

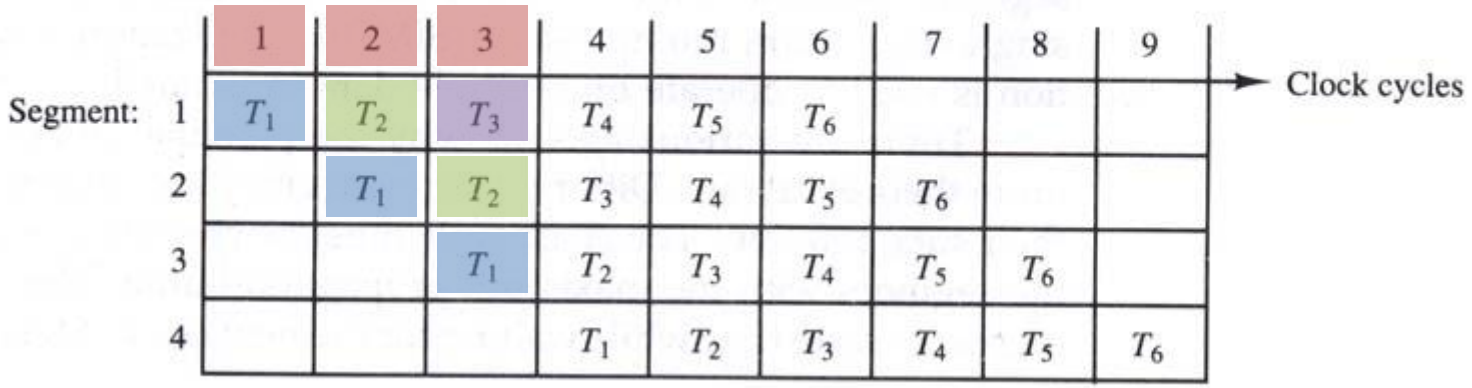


Six tasks T_1 to T_6 execute in four segments

Pipelining

- General considerations (cont.)
  - Space-time diagram

Figure 9-4 Space-time diagram for pipeline.



Six tasks T_1 to T_6 execute in four segments

Pipelining

- General considerations (cont.)
  - Space-time diagram

Figure 9-4 Space-time diagram for pipeline.

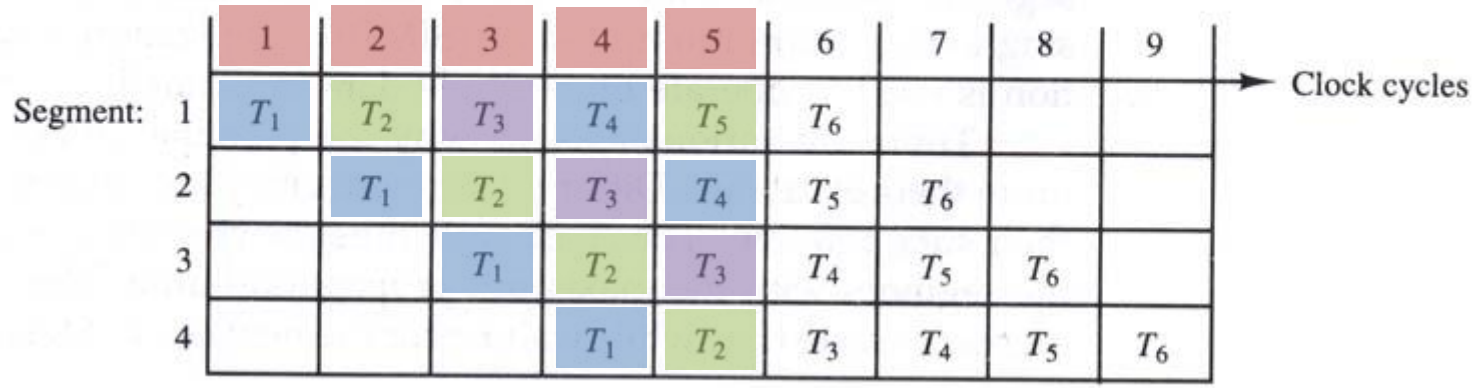
Complete: T_1

Six tasks T_1 to T_6 execute in four segments

Pipelining

- General considerations (cont.)
  - Space-time diagram

Figure 9-4 Space-time diagram for pipeline.



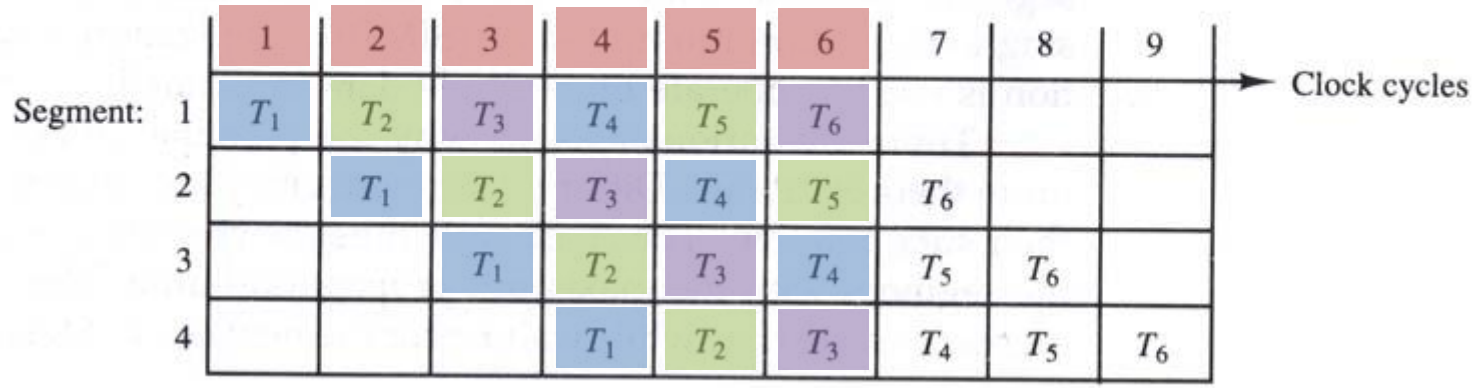
Complete: T_1 T_2

Six tasks T_1 to T_6 execute in four segments

Pipelining

- General considerations (cont.)
  - Space-time diagram

Figure 9-4 Space-time diagram for pipeline.



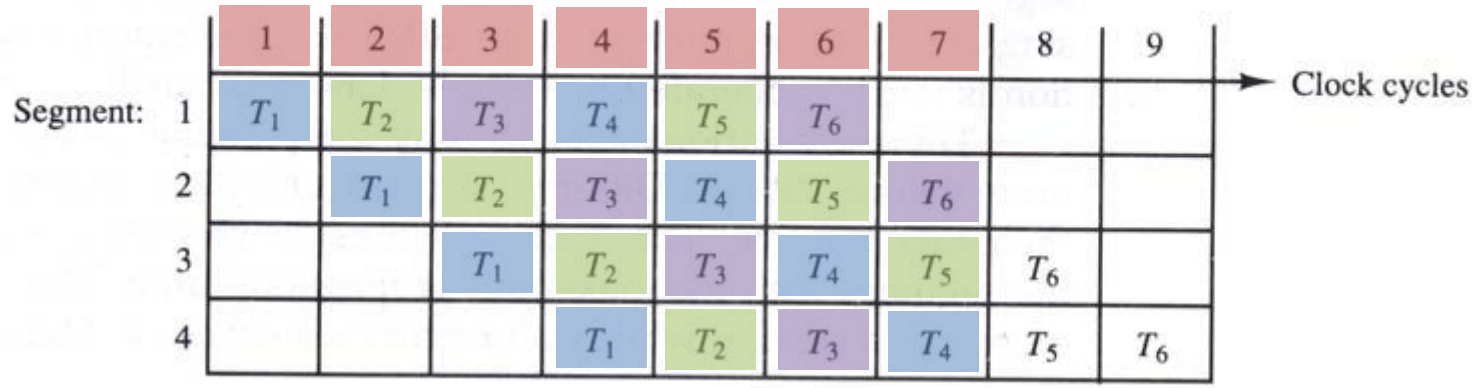
Complete: T_1 T_2 T_3

Six tasks T_1 to T_6 execute in four segments

Pipelining

- General considerations (cont.)
  - Space-time diagram

Figure 9-4 Space-time diagram for pipeline.



Complete: **T_1 T_2 T_3 T_4**

Six tasks T_1 to T_6 execute in four segments

Pipelining

- General considerations (cont.)
  - Space-time diagram

Figure 9-4 Space-time diagram for pipeline.

Complete: **T_1 T_2 T_3 T_4 T_5**

Six tasks T_1 to T_6 execute in four segments

Pipelining

- General considerations (cont.)
  - Space-time diagram

Figure 9-4 Space-time diagram for pipeline.

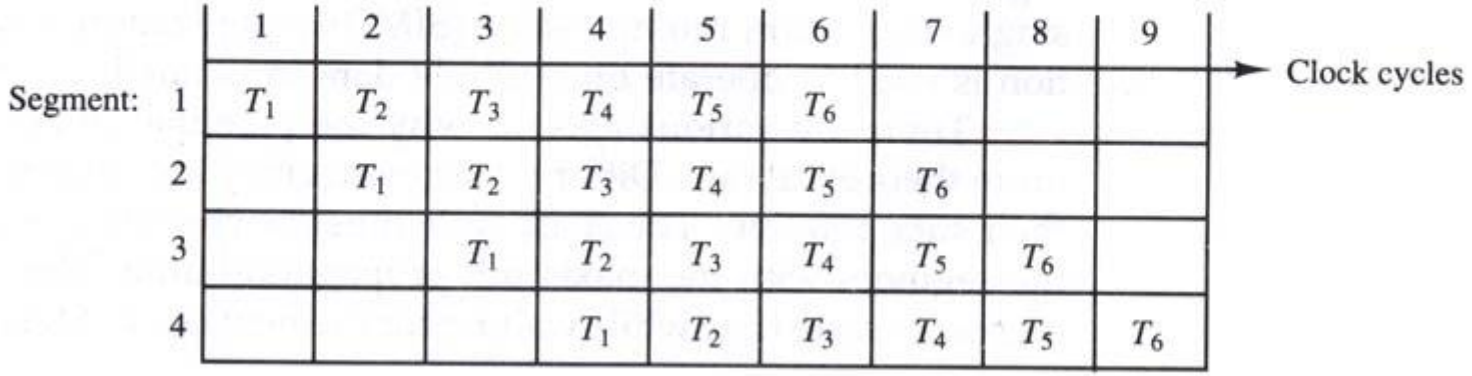
Complete: **T_1 T_2 T_3 T_4 T_5 T_6**

Six tasks T_1 to T_6 execute in four segments

Pipelining

- General considerations (cont.)
  - Space-time diagram

Figure 9-4 Space-time diagram for pipeline.



Six tasks T_1 to T_6 execute in four segments

- t_p: clock cycle time, k: # of segments, n: # of tasks
- First task T_1: kt_p, remaining (n-1) tasks T_2-T_6: $(n-1)t_p$
- To complete n tasks using k-segment: k + (n-1) clock cycles

Pipelining

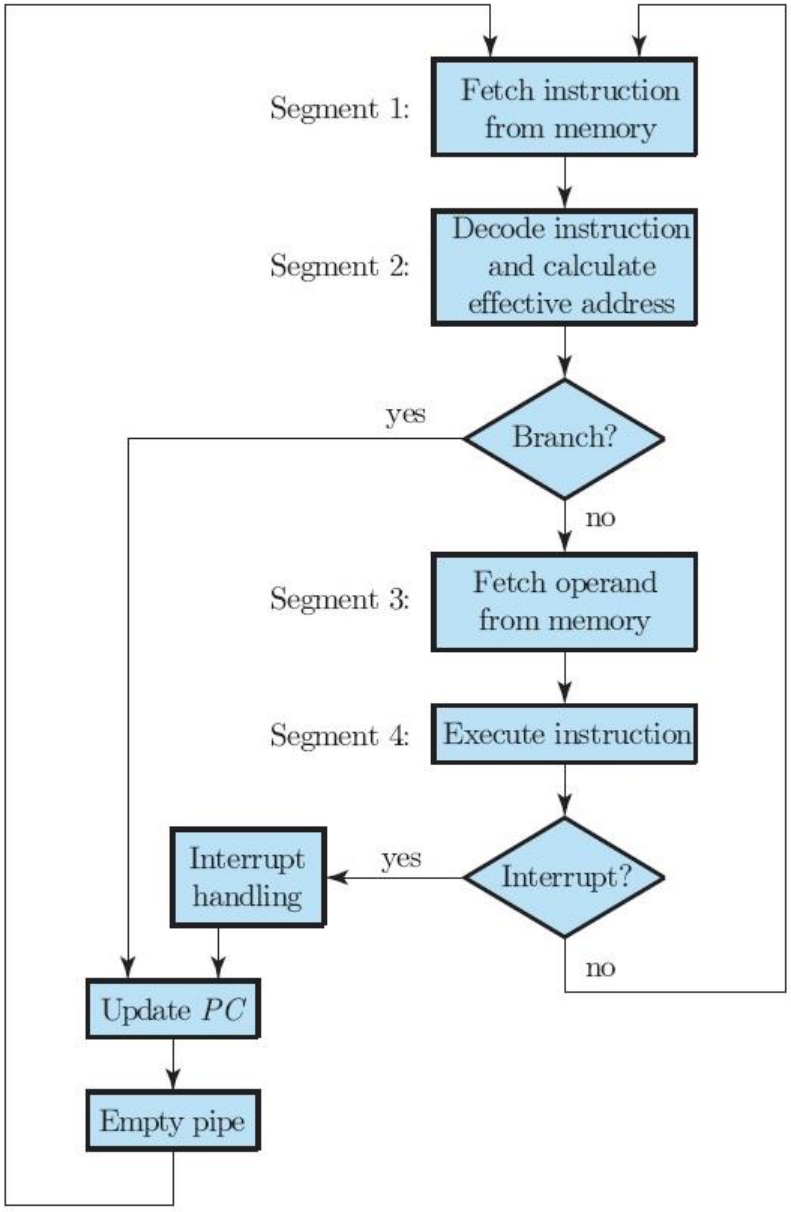
- General considerations (cont.)
  - Speedup

$$S = \frac{nt_n}{(k + n - 1)t_p}$$

    - t_n: consumed time for the same operation using non-pipeline
    - When n → ∞ $$S = \frac{t_n}{t_p}$$
    - If $t_n = kt_p$ $$S = \frac{kt_p}{t_p} = k$$; theoretical maximum speedup
  - Usually cannot operate at its maximum theoretical rate
    - Different segments take different time
    - Intermediate registers

Instruction pipeline

- 4-segment instruction pipeline
  - (1) FI: fetch an instruction from memory
  - (2) DA: decode the instruction and calculate the effective address of the operand
  - (3)FO: fetch the operands from memory
  - (4) EX: execute the operation



.kr

Instruction pipeline

- Timing of 4-segment instruction pipeline

Step:	1	2	3	4	5	6	7	8	9	10
1	FI	DA	FO	EX						
2		FI	DA	FO	EX					
3			FI	DA	FO	EX				
4				FI	DA	FO	EX			
5					FI	DA	FO	EX		
6						FI	DA	FO	EX	
7							FI	DA	FO	EX

Instruction pipeline

- Timing of 4-segment instruction pipeline

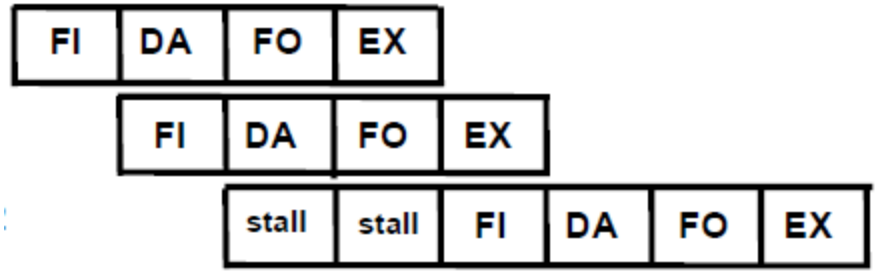
	Step:	1	2	3	4	5	6	7	8	9	10	11	12	13
Instruction:	1	FI	DA	FO	EX									
	2		FI	DA	FO	EX								
(Branch)	3			FI	DA	FO	EX							
	4				FI	-	-	FI	DA	FO	EX			
	5					-	-	-	FI	DA	FO	EX		
	6									FI	DA	FO	EX	
	7										FI	DA	FO	EX

Major hazards in pipelined execution

- Structural hazards (resource conflicts)
  - Hardware resources required by the instruction in simultaneous overlapped execution cannot be met
- Data hazards (data dependency conflicts)
  - An instruction scheduled to be executed in the pipeline requires the result of a previous instruction, which is not yet available
- Control hazards (branch address dependency conflicts)
  - Branches and other instructions that change the PC make the fetch of the next instruction to be delayed

Structural hazards

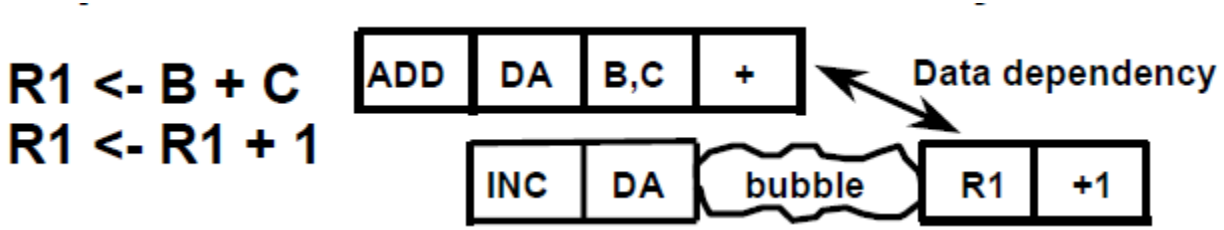
- Resource conflicts
- Occurs when some resource has not been duplicated enough to allow all combinations of instructions in the pipeline to execute
- Ex) with one memory-port, a data and an instruction fetch cannot be initiated in the same clock



  - Two-port memory will serve without stall

Data hazards

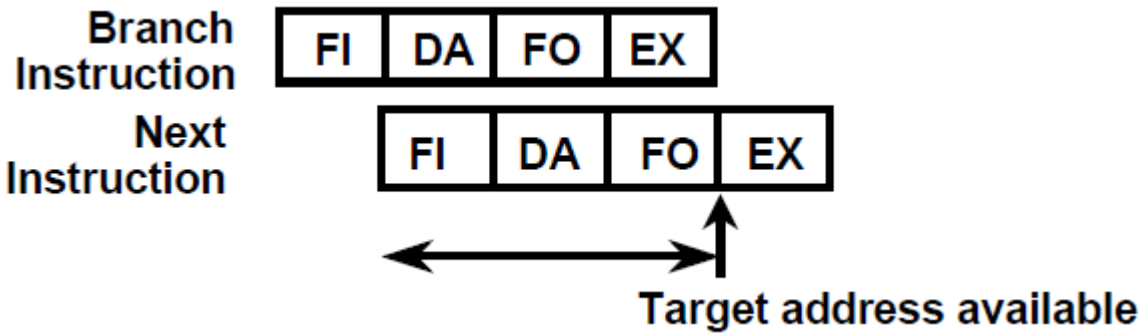
- Data dependency conflicts
- Occurs when the execution of an instruction depends on the results of a previous instruction



- Hardware technique
  - Hardware interlock
  - Operand forwarding
- Software technique
  - Instruction scheduling for “delayed load”

Control hazards

- Branch address dependency
- Branch target address is not known until the branch instruction is completed



- Dealing with control hazards
  - Pre-fetch target instruction
  - Branch target buffer
  - Loop buffer
  - Branch prediction
  - Delayed branch

RISC pipeline

- Three-segment instruction pipeline
- Data manipulation instructions
  - I: Instruction fetch
  - A: Decode, read registers, ALU operations
  - E: Write a register
- Data transfer instructions (load/store)
  - I: Instruction fetch
  - A: Decode, evaluate effective address
  - E: Register-to-memory or memory-to-register
- Program control instructions
  - I: Instruction fetch
  - A: Decode, evaluate branch address
  - E: write register (PC)

RISC pipeline

- Delayed load

LOAD: R1 ← M[address 1]
LOAD: R2 ← M[address 2]
ADD: R3 ← R1 + R2
STORE: M[address 3] ← R3

Clock cycles:	1	2	3	4	5	6
1. Load *R1*	I	A	E			
2. Load *R2*		I	A	E		
3. Add *R1*+*R2*			I	A	E	
4. Store *R3*				I	A	E

(a) Pipeline timing with data conflict

Clock cycles:	1	2	3	4	5	6	7
1. Load *R1*	I	A	E				
2. Load *R2*		I	A	E			
3. No-operation			I	A	E		
4. Add *R1*+*R2*				I	A	E	
5. Store *R3*					I	A	E

(b) Pipeline timing with delayed load

RISC pipeline

- Pipeline timing with branch address conflict

Clock cycles:	1	2	3	4	5	6	7	8
1. Load	I	A	E					
2. Increment		I	A	E				
3. Add			I	A	E			
4. Subtract				I	A	E		
5. Brach to X					I	A	E	
6. Instruction in X						I	A	E

RISC pipeline

- Delayed branch using no-operation instructions

Clock cycles:	1	2	3	4	5	6	7	8	9	10
1. Load	I	A	E							
2. Increment		I	A	E						
3. Add			I	A	E					
4. Subtract				I	A	E				
5. Brach to X					I	A	E			
6. No-operation						I	A	E		
7. No-operation							I	A	E	
8. Instruction in X								I	A	E

(a) Using no-operation instructions

RISC pipeline

- Delayed branch by rearranging the instructions

Clock cycles:	1	2	3	4	5	6	7	8
1. Load	I	A	E					
2. Increment		I	A	E				
3. Branch to X			I	A	E			
4. Add				I	A	E		
5. Subtract					I	A	E	
6. Instruction in X						I	A	E

(b) Rearranging the instructions

Problems

- 9-1, 9-2, 9-4, 9-12, 9-13