Chapter 4

Instruction Pipelining

LEARNING OBJECTIVES

- *Flynn's classification*
- *Pipelining*
- *Six-stages of pipelining*
- *Pipeline performance*
- *Pipeline hazards*
- *Structural hazards*
- *Data hazards*
- *Control hazards*
- *Conditional branch*
- *Dealing with branches*

Flynn's Classification

In parallel processing, the system is able to perform concurrent data processing to achieve faster execution time. A classification introduced by M.J. Flynn considers the organization of a computer system by the number of instructions and data items that are manipulated simultaneously. According to this classification, there are four major groups of computers:

1. **Single instruction stream, single data stream (SISD):**
   - Single computer containing a control unit, a processor unit and a memory unit
   - Instructions executed sequentially; parallel processing may be achieved by multiple functional units or by pipeline processing.
2. **Single instruction stream, Multiple data stream (SIMD):**
   - Include many processing units under the supervision of a common control unit.
   - All processors receive the same instruction from the control unit but operate on different items of data.
3. **Multiple instruction stream, Single data stream (MISD):**
   - No practical system has been constructed.
4. **Multiple instruction stream, Multiple data stream (MIMD):**
   - Capable of processing several programs at the same time.

One type of parallel processing that does not fit Flynn's classification is pipelining.

Pipelining

Pipelining is a technique of decomposing a sequential process into sub operations, with each subprocess being executed in special dedicated segment that operates with all other segments.

In pipelining, new inputs are accepted at one end before previously accepted inputs appear as outputs at the other end.

Two-stage Pipeline

As a simple approach, consider subdividing instruction processing into two stages:

1. Fetch instruction
2. Execute instruction
   - There are times during the execution of an instruction when main memory is not being accessed. This time can be used to fetch the next instruction in parallel with the execution of the current one. This is called instruction prefetch or fetch overlap.
   - This process will speed up instruction execution. If the fetch and execute stages were of equal duration, the instruction cycle time would be halved.
   - But there are some problems in this technique:
     (i) The execution time is generally longer than the fetch time.
     (ii) A conditional branch instruction makes the address of the next instruction to be fetched unknown.
   - These two factors reduce the potential effectiveness of the two-stage pipeline, but some speed up occurs. To gain further speed up, the pipeline must have more stage(s).

Six-stage Pipeline

Let the six stages be

F: Fetch Instruction
D: Decode Instruction
C: Calculate Operand Address
O: Operand Fetch

E: Execute instruction
W: Write operand

Then the time line diagram for seven instructions is shown below:

Clock cycle	1	2	3	4	5	6	7	8	9	10	11	12
Instruction *i*	F	D	C	O	E	W						
i+1		F	D	C	O	E	W					
i+2			F	D	C	O	E	W				
i+3				F	D	C	O	E	W			
i+4					F	D	C	O	E	W		
i+5						F	D	C	O	E	W	
i+6							F	D	C	O	E	W

Execution Time for the seven instructions with pipelining $= \left(\frac{t_{ex}}{6}\right) \times 12 = 2 * t_{ex}$. Where t_{ex} in the execution time required for each instruction.

- A deeper pipeline means that there are more stages in the pipeline. This generally means that the processor's frequency can be increased as the cycle time is lowered. This happen because there are fewer components in each stage of the pipeline, so the propagation delay is decreased for the overall stage.
- An instruction pipeline is said to be fully pipelined if it can accept a new instruction in every clock cycle.
- A pipeline that is not fully pipelined has wait cycle that delays the progress of the pipeline.

Advantages of pipelining:
- The cycle time of the processor is reduced, thus increasing instruction issue rate in most cases.
- Some combinational circuits such as adders or multipliers can be made faster by adding more circuitry. If pipelining is used instead it can save circuitry versus a more complex combinational circuit.

Limitations of pipelining:
1. If the stages are not of equal duration, there will be some waiting involved at various stages.
2. Conditional branch instruction may invalidate several instruction fetches.
3. The contents of one stage may depend on the contents of other stages of previous instructions, which is still in pipeline.

Pipeline Performance

The cycle time τ of an instruction pipeline is the time needed to advance a set of instructions one stage through the pipeline.

$$\text{Cycle time} = \max[\tau_i] + d = \tau_m + d,\ 1 \le i \le K$$

where τ_i = Time delay of the circuitry in the i^{th} stage of the pipeline.

τ_m = maximum stage delay.
K = number of stages in instruction pipeline
d = time delay of a latch, needed to advance signals and data from one stage to the next.

Suppose that n instructions are processed without any branches. Let $T_{k,n}$ be the total time required for a pipeline with K stages to execute n instructions. Then

$$T_{k,n} = [K+(n-1)]\tau$$

Example 1: Let $n = 7, K = 6, \tau = 1$. Then $T_{k,n} = [6 + (7 - 1)] \times 1 = 12$ cycles.

Now consider a processor with equivalent functions but no pipeline and assume that the instruction cycle time is $k\tau$. The speed up factor for the instruction pipeline compared to execution without the pipeline is defined as

$$S_k = \frac{T_{1,n}}{T_{k,n}} = \frac{nk\tau}{[(k+(n-1)]\tau} = \frac{nk}{k+(n-1)}$$

Note: Larger the number of stages, greater the potential for speed up. But practically, the potential gains of additional pipeline stages are countered by increase in cost, delays between stages, and the fact that braches will be encountered requiring the flushing of the pipeline.

Arithmetic pipeline:
- Pipeline arithmetic units are usually found in very high-speed computers.
- These are used to implement floating point operations, multiplication of fixed point numbers and similar computations encountered in scientific problems.

Pipeline Hazards

- Pipeline hazards are situations that prevent the next instruction in the instruction stream from executing during its designated clock cycle. The instruction is said to be stalled. When an instruction is stalled, all instructions later in the pipeline than the stalled instructions are also stalled. Instructions earlier than the stalled one can continue. No new instructions are fetched during the stall.

Note: Keeping a pipeline at its maximal rate is prevented by pipeline hazard.

Different types of Hazards:

1. Structural Hazards
2. Data Hazards
3. Control Hazards

Structural Hazards

Structural hazards occur when a certain resource is requested by more than one instruction at the same time.

Example 2: Instruction MVI *B*, *X* fetches in the *O* stage operand *X* from memory. The memory does not accept another access during that cycle.

Clock cycle	1	2	3	4	5	6	7	8	9	10	11
MVI *B*, *X*	F	D	C	O	E	W					
Instruction $i+1$		F	D	C	O	E	W				
$i+2$			F	D	C	O	E	W			
$i+3$				stall	F	D	C	O	E	W	
$i+4$						F	D	C	O	E	W

Penalty: 1 cycle

Certain resources are duplicated in order to avoid structural hazards (ALU, floating-point unit) can be pipelined themselves in order to support several instructions at a time. A classical way to avoid hazards at memory access is by providing separate data and instruction caches.

Note: Structural hazards are due to resource conflict.

Data Hazards

In a pipeline execution of two instructions I_1 and I_2 a certain stage of the pipeline I_2 needs the result produced by I_1, but this result has not yet been generated, then we have a data hazard.

Example 3: I_1: ADD R_3, R_2 $\quad R_3 \leftarrow R_3 + R_2$

I_2: MUL R_1, R_3 $\quad R_1 \leftarrow R_1 * R_3$

Clock cycle	1	2	3	4	5	6	7	8	9	10	11	12
ADD R_3, R_2	F	D	C	O	E	W						
MUL R_1, R_3		F	D	C	stall	stall	O	E	W			
Instruction $i+2$			F	D			C	O	E	W		

Penalty: 2 cycles Before executing the O stage (operand fetch stage), the MUL instruction is stalled until the ADD instruction has written the result into R_3.

Data dependencies

Data dependency exists between two instructions if the data data used by an instruction depends on the data created by other instructions.

Two type of dependencies exist between instructions:

1. True data dependency
2. Name dependencies
   (i) Anti-dependency
   (ii) Output dependency

True data dependency

- This is also called as Read-After-Write Hazard (RAW).
- This type of dependency occurs when the value produced by an instruction is required by a subsequent instruction.
- This is also known as a flow dependency because dependency is due to flow of data in a program.

Example: ADD R_3, R_2, R_1; $R_3 \leftarrow R_2 + R_1$

SUB R_4, R_3, 1; $R_4 \leftarrow R_3 - 1$

- Here R_3 is read before it is written by 'ADD' instruction.
- In RAW hazard $(i+1)^{st}$ instruction tries to read a source before it is written by 'i^{th}' instruction. So $(i+1)^{st}$ instruction incorrectly gets the old value.
- This kind of hazard can be reduced by using forwarding (or Bypassing).

Name dependencies

1. Anti-dependency:

- This is also called as Write-After-Read hazard
- This kind of dependency occurs when an instruction writes to a location which has been read by a previous instruction.
- Here $(i+1)^{st}$ instruction tries to write an operand before it is read by i^{th} instruction. So i^{th} instruction incorrectly gets the new value.

Example: I_1: ADD R_3, R_2, R_1 ;$R_3 \leftarrow R_2 + R_1$

I_2: SUB R_2, R_5, 1; $R_2 \leftarrow R_5 - 1$

I_2 must not produce its result in R_2 before I_1 read R_2, otherwise I_1 would use the value produced by I_2 rather than the previous value of R_2.

2. Output dependency:

- This is also called as Write - After - Write (WAW) hazard.
- This dependency occurs when a location is written by two instructions.
- i.e., $(i+1)$th instruction tries to write an operand before it is written by i^{th} instruction.

The writes end up being performed in the wrong order.

Example: I_1: ADD R_3, R_2, R_1;$R_3 \leftarrow R_2 + R_1$

I_2: SUB R_2, R_3, 1; $R_2 \leftarrow R_3 - 1$

I_3: R_3, R_2, R_5; $R_3 \leftarrow R_2 + R_5$

There is a possibility of WAW hazard between I_1 and I_3.

Handling data dependency There are ways to handle date dependency.

1. Hardware interlocks
2. Operand forwarding
3. Delayed load

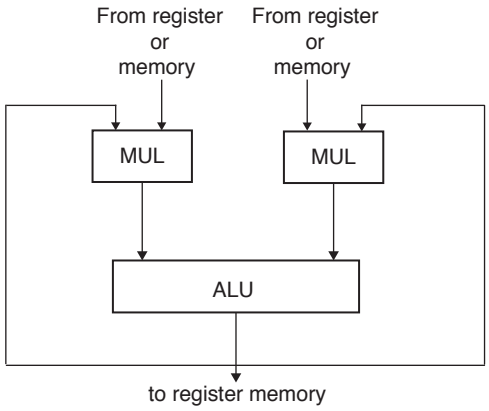
1. Hardware interlocks:

- To avoid data dependency, insert hardware interlock.
- An interlock is a circuit that detects instructions whose source operands are destinations of instructions farther up in the pipeline.

2. Operand forwarding:

- Uses special hardware to detect a conflict and then avoid it by routing the data through special paths between pipeline segments.

- Some of the penalty produced by data hazards can be avoided using a technique called forwarding (Bypassing).
- The ALU result is always fed back to the ALU input. If the hardware detects that the value needed for the current operation is the one produced by the previous operation (but which has not yet been written back). It selects the forwarded result as the ALU input, instead of the value read from register or memory.



Clock cycle	1	2	3	4	5	6	7	8
ADD R_3, R_2	F	D	C	O	E	W		
MUL R_1, R_3		F	D	C	stall	O	E	W

Penalty: 1 cycle After the E stage of the MUL instruction the result is available by forwarding. Therefore the penalty is reduced to one cycle.

Delayed Load: Here the compiler of a computer will detect the data conflicts and reorder the instructions as necessary to delay the loading of the conflicting data by inserting no-operation instruction.

Control Hazards

Control hazards are produced by Branch Instructions.

Unconditional branch

- Jump loop
- Loop

Clock cycle	1	2	3	4	5	6	7	8	9	10	11
Loop	F	D	C	O	E	W					
Loop		stall	stall	stall	F	D	C	O	E	W	
Loop + 1						F	D	C	O	E	W

Penalty: 3 cycles

- The instruction following the branch is fetched before the D stage is finished in 2nd clock. It is not known that a branch is executed. Later the fetched instruction is discarded.
- After the O stage of the branch instruction the address of the target is known and it can be fetched.

Conditional branch

Example: ADD B; $A \leftarrow A + B$
JZ Loop
Loop: If condition satisfies and branch is taken:

Clock cycle	1	2	3	4	5	6	7	8	9	10	11	12
ADD *B*	F	D	C	O	E	W						
JZ Loop		F	D	C	O	E	W					
Loop			stall	stall	stall	F	D	C	O	E	W	

Penalty:3 cycles

At this moment both the condition (set by ADD) and the target address are known.

If condition not satisfied and branch not taken:

Clock cycle	1	2	3	4	5	6	7	8	9	10	11	12
ADD *B*	F	D	C	O	E	W						
JZ Loop		F	D	C	O	E	W					
Instruction *i* + 1			F	stall	stall	D	C	O	E	W		

Penalty:2 cycles

At this moment the condition is known and instruction i + 1 can go on.

- With conditional branch, we have a penalty even if the branch has not been taken. This is because we have to wait until the branch condition is available.

Dealing with branches

One of the major problems in designing an instruction pipeline is the occurrence of branch instructions. A variety of approaches have been taken for dealing with braches

1. Multiple streams
2. Prefetch branch target
3. Branch target buffer
4. Loop buffer
5. Branch prediction
6. Delayed branch

Multiple streams A branch instruction may cause to choose one of two instructions to fetch next, then allow the pipeline to fetch both instructions, making use of streams.

There are two problems with this approach:

1. With multiple pipelines there are contention delays for access to the registers and to memory.
2. Additional branch instructions may enter the pipeline before the original branch decision is resolved.

Prefetch branch target When a conditional branch is recognized, the target of the branches is prefetched, in addition to the instruction following the branch. This target is then saved until the branch instruction is executed.

Branch targets buffer (BTB) BTB is an associative memory included in the fetch segment of the pipeline. Each entry in the BTB consists of the address of a previously executed branch instruction and target instructions for that branch.

It also stores the next few instructions after the branch target instruction.

When the pipeline decodes a branch instruction, it searches the associative memory BTB for the address of the instruction. If it is in BTB, the instruction is available directly and prefetch continues from the new path. If the instruction is not in BTB, the pipeline shifts to a new instruction stream and stores the target instruction in the BTB.

Advantage: Branch instructions that occurred previously are readily available in the pipeline without interruption.

Loop Buffer A loop buffer is a small, very high speed memory maintained by the instruction fetch stage of the pipeline and containing the n most recently fetched instructions in sequence. If a branch is to be taken, the hardware first checks whether the branch target is within the buffer. If so, the next instruction is fetched from the buffer. The Advantages of loop buffer are

1. Loop buffer will contain some instructions sequentially ahead of the current instruction fetch address. Thus instructions fetched in sequence will be available without the usual memory access time.
2. If the branch occurs to a target just a few locations ahead of the address of the branch instruction, the target will already be in the buffer.
3. This strategy is well suited in dealing with loops.

Branch prediction Various techniques can be used to predict whether a branch will be taken or not. The common techniques are

Static { 1. Predict never taken Static
2. Predict always taken
3. Predict by opcode

Dynamic { 4. Taken/not taken switch
5. Brach history table

- The first two approaches are static, i.e., no dependency on execution history. Here always assume that the branch will not be taken and continue to fetch instructions in sequence, or always assume that the branch will be taken and always fetch from the branch target.
- The third approach is also static. Takes the decision based on the opcode of the branch instruction in a program.
- Dynamic branch strategies attempt to improve the accuracy of prediction by recording the history of conditional branch instructions in a program.

(a) Taken/not taken switch:

- Use two bits to record the result of the last two instances of the execution of the associated instruction or record a state in some other fashion.

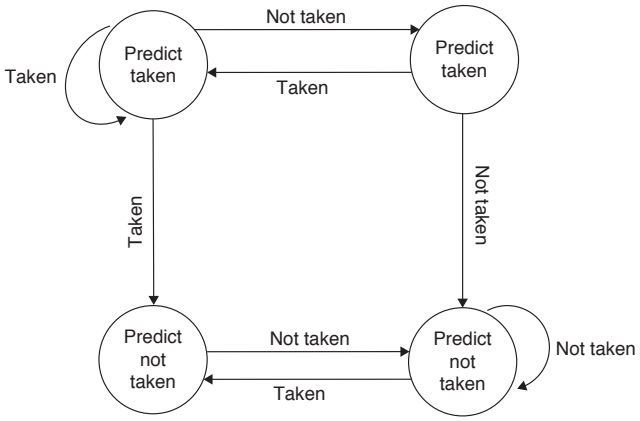


Figure 1 Branch prediction state diagram

- As long as each succeeding conditional branch instruction that is encountered is taken, the decision process predicts that the next branch will be taken.
- If a single prediction is wrong, the algorithm continues to predict that the next branch is taken.
- Only if two successive branches are not taken does the algorithm shift to not taken branch.

Drawback: If the decision is made to take the branch, the target instruction cannot be fetched until the target address, which is an operand in the conditional branch instruction is decoded.

(b) Branch history table: It is a small cache memory associated with the instruction fetch stage of the pipeline.

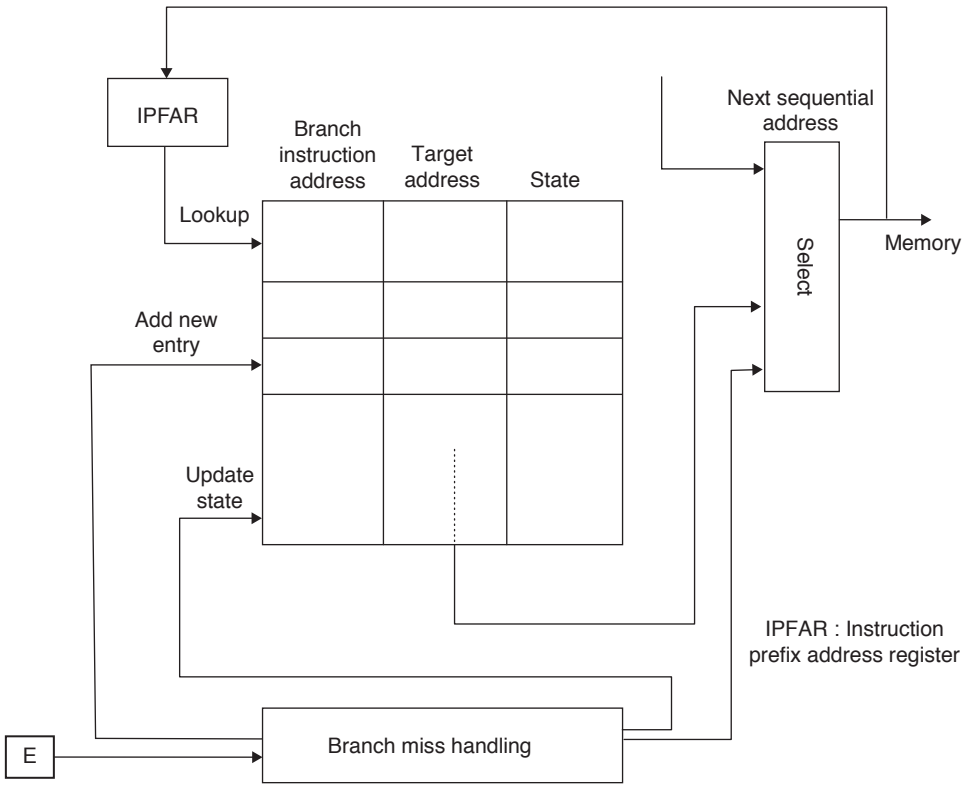


Figure 2 Branch history table

- Each entry in the table consist of three elements:
  1. The address of branch instruction.
  2. Some number of history bits that record the state of use of that instruction.
  3. Information about target instruction.

Delayed branch A complier detects the branch instructions and rearranges the machine language code sequence by inserting useful instructions that keep a pipeline operating without interruptions.

Exercises

Practice Problems I

Directions for questions 1 to 21: Select the correct alternative from the given choices.

Common data for questions 1 and 2: An unpipelined processor with eight number cycle time and pipeline batches with 1 ns latency is given.

1. Find the cycle times of pipelined versions of the processor with 2, 4, 8 and 16 stages if the Data path logic is evenly divided among the pipeline stages.
(A) 5, 3, 2, 1.5 (B) 4, 2, 1, 0.5
(C) 8, 4, 2, 1 (D) 10, 6, 4, 3

2. What is the latency of each of the pipelined versions of the processor?
(A) 4, 2, 1, 0.5 (B) 10, 6, 4, 3
(C) 5, 3, 2, 1.5 (D) 10, 12, 16, 24

3. A 4-stage pipeline has the stage delays as 110, 120, 130, and 140 nanoseconds respectively. Registers that are used between the stages have a delay of 2 nanoseconds each. Assuming constant clocking rate. Find the total time taken to process 1000 instructions on this pipeline.
(A) 7.1 ms (B) 14.24 ms
(C) 28 ms (D) 2000 ms

4. Consider a pipelined processor with the following four stages:
IF: instruction fetch
ID: Instruction decode
EX: Execute
WB: Write back
The IF, ID and WB stages takes one clock cycle each to complete the operation. The number of clock cycles for EX stage depends on the instruction; for I_1 and I_3 one clock cycle is needed and for I_2 three clock cycles are needed. Find the number of clock cycles taken to complete the following sequence of instructions?

I_1:	ADD R_0,	R_1,	R_2	$R_0 \leftarrow R_1 + R_2$
I_2:	MUL R_2,	R_3,	R_4	$R_2 \leftarrow R_3 \times R_4$
I_3:	SUB R_4,	R_5,	R_6	$R_4 \leftarrow R_5 - R_6$

(A) 7 (B) 8
(C) 6 (D) 9

5. A CPU has five stage pipelines and runs at 1 GHz frequency. Instruction fetch happens in the first stage of the pipeline. A conditional branch instruction computes the target address and evaluates the condition in the third stage of the pipeline. The processor stops fetching new instructions following a conditional branch until the branch outcome is known. A program executes 10^9 instructions. Out of which 10% are conditional branches. If each instruction takes one cycle to complete on average then find the total execution time of the program?
(A) 1 sec (B) 1.2 sec
(C) 1.4 sec (D) 1.8 sec

6. Consider a four stage pipeline processor, number of cycles needed by the four instructions I_1, I_2, I_3 and I_4 in stages S_1, S_2, S_3 and S_4 are shown below:

	S_1	S_2	S_3	S_4
I_1	2	1	1	1
I_2	1	3	2	2
I_3	2	1	1	3
I_4	1	2	2	2

What is the number of cycles needed to execute the instructions in the order:
$I_1 : I_2 : I_3 : I_4$
(A) 8 (B) 12
(C) 14 (D) 15

7. A non-pipelined system takes 50 ns to process a task; the same task can be processed in a six-segment pipeline with a clock cycle of 10 ns. Speedup ratio of 100 tasks for pipeline is
(A) 1.62 (B) 3.21
(C) 4.76 (D) 8.21

8. Consider a pipelined processor with the following four stages:
IF: Instruction fetch
ID: Instruction decode
EX: Execute
WB: Write back

The IF, ID and WB stages takes one clock cycle each to complete the operation. The number of clock cycles for EX stage depends on the instruction; for I_1 and I_3 one clock cycle is needed and for I_2 three clock cycles are needed. The number of clock cycles taken to complete the following sequence of instructions is

I_1:	ADD R_0, R_1, R_2,	$R_0 \leftarrow R_1 + R_2$
I_2:	MUL R_2, R_0, R_4,	$R_2 \leftarrow R_0 \times R_4$
I_3:	SUB R_4, R_5, R_2,	$R_4 \leftarrow R_5 - R_2$

(A) 7 (B) 8
(C) 9 (D) 10

9. Following are the sequence of stages in a pipeline CPU:
(1) IF: Instruction fetch from instruction memory
(2) RD: Instruction decode and register read
(3) EX: Execute ALU operation for data and address computation
(4) MA: Data memory access, for write access, the register read at RD stage is used.
(5) WB: Register write back

Consider the following sequence of instructions:
LOAD R_1, M[loc]
ADD R_1, R_1, R_1
ADD R_2, R_1, R_2
Let each stage take one clock cycle.

What is the number of clock cycles taken to complete the above sequence of instructions starting from the fetch of first instruction?
(A) 18 (B) 15
(C) 13 (D) 10

10. Which of the following can cause a hazard for a pipelined CPU with a single ALU?
(i) The $(j + 1)^{st}$ instruction uses the result of the j^{th} instruction as an operand.
(ii) The j^{th} and $(j + 1)^{st}$ instructions require the ALU at the same time.
(iii) The execution of a conditional jump instruction.
(iv) The execution of non-conditional jump instruction.
(A) (i) and (ii) (B) (ii) and (iii)
(C) (i), (ii) and (iii) (D) (i), (ii), (iii) and (iv)

11. Given an unpipelined processor with a 10 ns cycle time and pipeline latches with 0.5 ns latency, how many stages of pipelining are required to achieve cycle time of 2 ns?
(A) 5.5 (B) 6.67
(C) 7 (D) 6

12. In a 4-stage pipeline,
IF – instruction fetches
ID – instruction decode and fetch operands
EX – Execute
WB – write back
ADD, SUB take one clock cycle, MUL take three clock cycles. Then for
ADD R_2, R_1, R_0 $R_2 \leftarrow R_1 + R_0$
MUL R_4, R_3, R_2 $R_4 \leftarrow R_3 * R_2$
SUB R_6, R_5, R_4 $R_6 \leftarrow R_5 - R_4$
Number of clock cycles required using operand forwarding technique are
(A) 8 (B) 12
(C) 10 (D) 14

13. Consider an instruction sequence of length 'n' that is streaming through a K-stage instructions pipeline. Let P be the probability of encountering a conditional or

unconditional branch instruction and let q be the probability that execution of a branch instruction I_B causes a jump to a non-consecutive address. Assume that each such jump requires the pipeline to be cleared, destroying all ongoing instruction processing, when I_B emerges from the last stage. Also assume that T is the cycle time. Then which of the following expression correctly specifies the time required for this pipeline?

(A) $pqnk\tau + (1 - pq)\,[K + (n - 1)]\tau$
(B) $(1 - pq)\,[k + (n - 1)]\tau + pqn\tau$
(C) $pqnk\tau + (1 - pq)\,n[k + (n - 1)]\tau$
(D) $pqn + (1 - pq)n\,[k + (n - 1)]\tau$

14. If T_m is maximum stage delay of an m-stage pipeline with time delay of the latch is d then cycle time is

(A) T_m/d (B) $T_m + d$
(C) $2T_m + d$ (D) $T_m \times d$

15. Pipelining is a general technique for increasing processor ____ without requiring large amounts of extra hardware.

(A) turnaround time (B) waiting time
(C) latency (D) throughput

16. A 4-stage instruction pipeline executes a 100 instruction program. The probability of occurrence of a conditional or unconditional branch is 0.4 and the probability of execution of a branch instruction I_B causing a jump to a non-consecutive address is 0.1. Then the speed up factor for the instruction pipeline compared to execution without pipeline is

(A) 2.14 (B) 6.23
(C) 3.21 (D) 3.48

17. A non-pipelined processor has a clock rate of 2.5GHz and an average cycles per instruction of 4. An upgrade to the processor introduces a five stage pipeline. However, due to internal pipeline delays, such as latch delay, the clock rate of the new processor has to be reduced to 2GHz. What is the MIPS rate for each of these processors respectively.

(A) 625, 400 MIPS (B) 625, 2000 MIPS
(C) 3125, 2000 MIPS (D) 3125, 400 MIPS

18. Consider the following sequence of instructions:

I_1: MUL R_1, R_2 $R_1 \leftarrow R_1 * R_2$
I_2: SUB $R_3, 1$ $R_3 \leftarrow R_3 - 1$
I_3: ADD R_3, R_4 $R_3 \leftarrow R_3 + R_4$
I_4: BEZ Target Branch if zero
I_5: MOVE R_3,10 $R_3 \leftarrow 10$
⋮

Target:

Which of the following instruction will be placed in delayed slot to reduce penalty in a 6-stage pipeline? (Assume that the branch outcome will be known during 5th stage)

(A) I_1 (B) I_2
(C) I_3 (D) I_5

19. Consider the following sequence of instructions:

ADD R_1, R_2 $R_1 \leftarrow R_1 + R_2$
BEZ Target Branch if Zero
MUL R_3, R_4 $R_3 \leftarrow R_3 * R_4$
MOVE R_1,10 $R_1 \leftarrow 10$
⋮

Target:

Assume that this program executed on a 6-stage pipelined processor and each stage required 1 clock cycle.

Let us suppose that "branch not taken" Prediction is used but the prediction is not fulfilled, then the penalty will be (branch outcome is known at 5th stage)

(A) 1 clock cycle (B) 2 clock cycles
(C) 3 clock cycles (D) 4 clock cycles

20. Suppose 40% of the instructions are loads and half the time they are followed by instruction that depends on value loaded. If this hazard causes single cycle delay, how must faster is ideal pipelined machine (CPI = 1) than real one? (Ignore other stalls)

(A) 1 time (B) 1.2 times
(C) 1.5 times (D) 1.15 times

21. Assume that a pipelined processor has three categories of instructions: Branch, load/store, other. If it is a branch instruction it will take 3 clock cycles, if it is a load/store instruction it will take 4 clock cycles and all other instructions require 6 clock cycles. A program consisting of 10% Branch instructions, 10% of load/store instructions is executed on this processor. Then the number of clock cycles required for the execution of the program is

(A) 2.45 (B) 3.61
(C) 4.66 (D) 5.5

Practice Problems 2

Directions for questions 1 to 20: Select the correct alternative from the given choices.

1. The time required for the five functional units, which operated in each of the five cycles are 10 ns, 7 ns, 10 ns, 10 ns and 8 ns. Assume that pipelining add 1 ns of overhead. The speed up of pipeline compared to unpipeline is

(A) 4.5 times (B) 1.1 times
(C) 4.1 times (D) 2.4 times

2. Which of the following is a technique of decomposing a sequential process into sub operations with each subprocess being executed in a special dedicated segment that operates concurrently with each other?

(A) Straight line sequencing
(B) Random sequencing
(C) Pipelining
(D) Serial execution

3. Which of the following statements is incorrect?
(A) Latency is the number of time units between two initiations in a pipelined architecture.
(B) If initiations are of different but fixed reservation tables, the architecture is known as static pipelined configuration.
(C) A collision in a pipelined architecture is an attempt by two different initiations to use the same stage at the same time.
(D) None of the above

4. Which of the following technique is used in a pipelined processor, when there is a conditional branch?
(A) Loop butter (B) Branch prediction
(C) Delayed Branch (D) All of the above

5. Which of the following cases, leads to a pipelined computer architecture?
(A) The evaluation of each basic function is relatively independent of the previous one.
(B) The sub-functions are closely related to each other.
(C) The evaluation of each sub function requires approximately the same sequence.
(D) All of the above

6. The performance of a pipelined processors is degraded if
(A) the pipeline stages have different delays.
(B) consecutive instructions are dependent on each other.
(C) the pipeline stages share hardware resources.
(D) All of the above

7. The following is a limit on how much the performance of a processor can be improved using pipelining:
(A) the number of pipeline stages
(B) data dependencies
(C) branch delays
(D) All of the above

8. A pipeline processor consists of a sequence of 'm' data processing circuits called ____, which collectively perform a single operation on a stream of data operands passing through them.
(A) stages (B) pipelines
(C) latches (D) None of the above

9. A five-stage pipelined CPU has the following sequence of stages:
IF – Instruction fetch from memory
RD – Decode instruction
EX – Execute
MA – Data memory access
WB – Register write back

Consider the following instruction sequence:
I_1: Load R_0 $R_0 \Leftarrow M$
I_2: ADD R_1, $R_1 R_1 \Leftarrow R_1 + R_1$
I_3: SUB R_2, $R_3 R_2 \Leftarrow R_2 - R_3$

Each stage takes one clock cycle.
Number of clock cycles to execute the program is
(A) 8 (B) 10
(C) 7 (D) 15

Common data for questions 10 and 11: Given an unpipelined processor with a 10 number cycle time and pipeline latches with 0.5 ns latency.

10. Which are the cycle times of pipelined versions of the processors with 2, 4, 8 and 16 stages if the data path logic is evenly divided among the pipeline stages?
(A) 5.0, 3.0, 1.5, 1.0
(B) 5.5, 3.0, 1.75, 1.125
(C) 4.0, 5.0, 6.0, 7.0
(D) None of the above

11. What is the latency of each of the pipelined versions of the processor with 2, 4, 8 and 16 stages?
(A) 10, 11, 12, 14 ns
(B) 10, 10, 11, 11 ns
(C) 11, 12, 14, 18 ns
(D) None of the above

12. Assume an unpipelined processor has a 1 ns clock cycle and it uses 5 cycles for ALU operations and branches. And 6 clock cycles for memory operations. A program has 40%, 30%, and 20% of ALU operations, branch instructions and memory operations respectively. If we are using pipelining it adds 0.2 ns overhead. Then what is the speedup of pipelining compared to unpipelined processor?
(A) 1.2 (B) 3.91
(C) 4.7 (D) 2.5

13. Consider a five-stage pipeline processor in which each instructions on an average has 2 clock cycle stalls. Then the speed up of this pipelined processor compared to an unpipelined processor is
(A) 2.5 (B) 1.67
(C) 0.4 (D) 5

14. Pipelining strategy is called to implement
(A) instruction execution
(B) instruction prefetch
(C) instruction decoding
(D) instruction manipulation

15. If an Instruction 'j' tries to read a source operand before instruction 'i' writes it. Then it is a ____ type of hazard.
(A) WAR (B) RAW
(C) WAW (D) None of these

16. What is the average instruction processing time of a five-stage instruction pipeline for 32 instructions if conditional branch instructions occur as follows: I_2, I_5, I_7, I_{25}, I_{27}.
(A) 1.97 (B) 1.67
(C) 1.75 (D) 1.25

17. Consider the execution of 1000 instructions on a five-stage pipeline machine. Then the speed-up due to the use of pipelining given that the probability of an instruction being a branch is 0.2.
(A) 1.77 (B) 2.6
(C) 2.77 (D) 3.2

18. If an instructions following a branch (taken or not taken) have a dependency on the branch and cannot be executed until the branch is executed, then the dependency is
(A) True data dependency
(B) Procedural dependency
(C) Resource conflict
(D) Output dependency

19. 'A two-stage instruction pipeline unlikely to cut the instruction cycle time in half, compared with the use of no pipeline.' The statement is
(A) Always true (B) Always False
(C) Can't predict (D) Some times true

20. Write after read dependency is also known as
(A) True dependency (B) Anti-dependency
(C) Output dependency (D) Inverse dependency

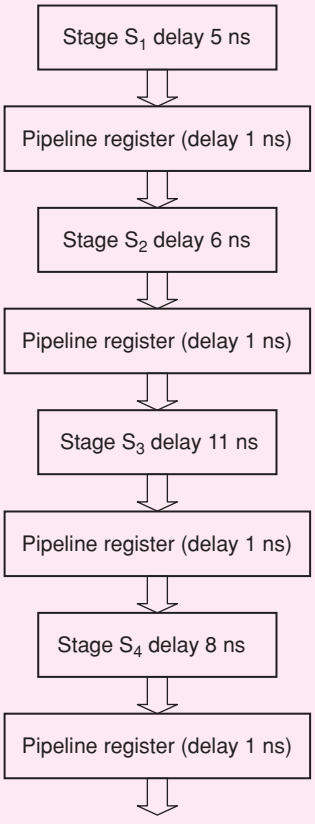
Previous Years' Questions

1. Consider a 6-stage instruction pipeline, where all stages are perfectly balanced. Assume that there is no cycle time overhead of pipelining. When an application is executing on this 6-stage pipeline, the speedup achieved with respect to non-pipelined execution if 25% of the instructions incur 2 pipeline stall cycles is _____. **[2014]**

2. Consider the following processors (ns stands for nano-seconds). Assume that the pipeline registers have zero latency.
P1: Four-stage pipeline with stage latencies 1 ns, 2 ns, 2 ns, 1 ns.
P2: Four-stage pipeline with stage latencies 1 ns, 1.5 ns, 1.5 ns, 1.5 ns.
P3: Five-stage pipeline with stage latencies 0.5 ns, 1 ns, 1 ns, 0.6 ns, 1 ns.
P4: Five-stage pipeline with stage latencies 0.5 ns, 0.5 ns, 1 ns, 1 ns, 1.1 ns.
Which processor has the highest peak clock frequency? **[2014]**
(A) P1 (B) P2
(C) P3 (D) P4

3. An instruction pipeline has five stages, namely, instruction fetch (IF), instruction decode and register fetch (ID/RF), instruction execution (EX), memory access (MEM), and register write back (WB) with stage latencies 1 ns, 2.2 ns, 2 ns, 1 ns, and 0.75 ns, respectively (ns stands for nano seconds). To gain in terms of frequency, the designers have decided to split the ID/RF stage into three stages (ID, RF1, RF2) each of latency 2.2/3 ns. Also, the EX stage is split into two stages (EX1, EX2) each of latency 1 ns. The new design has a total of eight pipeline stages. A program has 20% branch instructions which execute in the EX stage and produce the next instruction pointer at the end of the EX stage in the old design and at the end of the EX2 stage in the new design. The IF stage stalls after fetching a branch instruction until the next instruction pointer is computed . All instructions other than the branch instruction have an average CPI of one in both the designs. The execution times of this program on the old and the new design are P and Q nanoseconds, respectively. The value of P/Q is _____. **[2014]**

4. Consider an instruction pipeline with four stages (S_1, S_2, S_3 and S_4) each with combinational circuit only. The pipeline registers are required between each stage and at the end of the last stage. Delays for the stages and for the pipeline registers are as given in the figure.



What is the approximate speed up of the pipeline in steady state under ideal conditions when compared to the corresponding non-pipeline implementation? **[2011]**

(A) 4.0 (B) 2.5
(C) 1.1 (D) 3.0

5. Register renaming is done in pipelined processors **[2012]**

(A) as an alternative to register allocation at compile time
(B) for efficient access to function parameters and local variables
(C) to handle certain kinds of hazards
(D) as part of address translation

6. Consider an instruction pipeline with five stages without any branch prediction: Fetch Instruction (FI), Decode Instruction (DI), Fetch Operand (FO), Execute Instruction (EI) and Write Operand (WO). The stage delays for FI, DI, FO, EI and WO are 5 ns, 7 ns, 10 ns, 8 ns and 6 ns, respectively. There are intermediate storage buffers after each stage and the delay of each buffer is 1 ns. A program consisting of 12 instructions $I_1, I_2, I_3, \ldots, I_{12}$ is executed in this pipelined processor. Instruction I_4 is the only branch instruction and its branch target is I_9. If the branch is taken during the execution of this program, the time (in ns) needed to complete the program is **[2013]**

(A) 132 (B) 165
(C) 176 (D) 328

Common data for Questions 7 and 8: Delayed branching can help in the handling of control hazards

7. For all delayed conditional branch instructions, irrespective of whether the condition evaluates to true or false **[2008]**

(A) The instruction following the conditional branch instruction in memory is executed
(B) The first instruction in the fall through path is executed
(C) The first instruction in the taken path is executed
(D) The branch takes longer to execute than any other instruction

8. The following code is to run on a pipelined processor with one branch delay slot:

I_1: ADD $R_2 \leftarrow R_7 + R_8$
I_2: SUB $R_4 \leftarrow R_5 - R_6$
I_3: ADD $R_1 \leftarrow R_2 + R_3$
I_4: STORE Memory $[R_4] \leftarrow R_1$
BRANCH to Label if $R_1 = 0$

Which of the instructions I_1, I_2, I_3 or I_4 can legitimately occupy the delay slot without any other program modification? **[2008]**

(A) I_1 (B) I_2
(C) I_3 (D) I_4

9. A 5 stage pipelined CPU has the following sequence of stages:

IF - Instruction fetch from instruction memory

RD - Instruction decode and register read

EX - Execute: ALU operation for data and address computation

MA - Data memory access - for write access, the register read at RD state is used

WB - Register write back

Consider the following sequence of instructions.

I_1: $L\ R_0$, loc_1; $R_0 <= M[\text{loc}_1]$
I_2: $A\ R_0$, R_0; $R_0 <= R_0 + R_0$
I_3: $S\ R_2$, R_0; $R_2 <= R_2 - R_0$

Let each stage take one clock cycle.

What is the number of clock cycles taken to complete the above sequence of instructions from the fetch of I_1?

(A) 8 (B) 10
(C) 12 (D) 15

10. Consider a non-pipelined processor with a clock rate of 2.5 gigahertz and average cycles per instruction of four. The same processor is upgraded to a pipelined processor with five stages; but due to the internal pipeline delay, the clock speed is reduced to 2 gigahertz. Assume that there are no stalls in the pipeline. The speed up achieved in this pipelined processor is _____. **[2015]**

11. Consider the sequence of machine instructions given below:

MUL	R_5, R_0, R_1
DIV	R_6, R_2, R_3
ADD	R_7, R_5, R_6
SUB	R_8, R_7, R_4

In the above sequence, R_0 to R_8 are general purpose registers. In the instructions shown, the first register stores the result of the operation performed on the second and the third registers. This sequence of instructions is to be executed in a pipelined instruction processor with the following 4 stages: (1) Instruction Fetch and Decode (IF), (2) Operand Fetch (OF), (3) Perform Operation (PO) and (4) Write back the result (WB). The IF, OF and WB stages take 1 clock cycle each for any instruction. The PO stage takes 1 clock cycle for ADD or SUB instruction, 3 clock cycles for MUL instruction and 5 clock cycles for DIV instruction. The pipelined processor uses operand forwarding from the PO stage to the OF stage. The number of clock cycles taken for the execution of the above sequence of instructions is _____. **[2015]**

12. Consider the following reservation table for a pipeline having the stages S_1, S_2 and S_3.

Time →

	1	2	3	4	5
S_1	X				X
S_2		X		X	
S_3			X		

The minimum average latency (MAL) is _____

[2015]

13. Consider the following code sequence having five instructions l_1 to l_5. Each of these instructions has the following format. **[2015]**

OP Ri, *Rj*, *Rk*

Where operation *OP* is performed on contents of registers *Rj* and *Rk* and the result is stored in register *Ri*.

l_1: ADD R_1, R_2, R_3
l_2: MUL R_7, R_1, R_3
l_3: SUB R_4, R_1, R_5
l_4: ADD R_3, R_2, R_4
l_5: MUL R_7, R_8, R_9

Consider the following three statements.

S_1: There is an anti-dependence instructions between instructions l_2 and l_5

S_2: There is an anti-dependence between Instructions l_2 and l_4

S_3: |Within an instruction pipeline an anti-dependence always creates one or more stalls

Which one of the above statements is/are correct?
(A) Only S_1 is true
(B) Only S_2 is true
(C) Only S_1 and S_3 are true
(D) Only S_2 and S_3 are true

14. The stage delays in a 4 - stage pipeline are 800, 500, 400 and 300 picoseconds. The first stage (with delay 800 picoseconds) is replaced with a functionally equivalent design involving two stages with respective delays 600 and 350 picoseconds. The throughput increase of the pipeline is ___ percent. **[2016]**

15. Consider a 3 GHz (gigahertz) processor with a three - stage pipeline and stage latencies τ_1,τ_2 and τ_3 such that $\tau_1 = 3\tau_2/4 = 2\tau_3$. If the longest pipelines stage is split into two pipeline stages of equal latency, the new frequency is ____ GHz, ignoring delays in the pipeline registers. **[2016]**

16. Instruction execution in a processor is divided into 5 stages, *Instruction Fetch* (IF), *Instruction Decode* (ID), *Operand Fetch* (OF), *Execute* (EX), and *Write Back* (WB). These stages take **5, 4, 20, 10**, and **3 nanoseconds (ns)** respective. A pipelined implement action of the processor requires buffering between each pair of consecutive stages with a delay of **2 ns**. Two pipelined implementations of the processor are contemplated:

(i) A navie pipeline implementation (NP) with 5 stages and
(ii) An efficient pipeline (EP) where the OF stage is divided into stages OF1 and OF2 with execution times of **12 ns** and **8 ns** respectively.

The speedup (correct to two decimal places) achieved by EP over NP in executing 20 independent instructions with no hazards is _____. **[2017]**

17. The instruction pipeline of a RISC processor has the following stages: Instruction Fetch (IF), Instruction Decode (ID), Operand Fetch (OF), Perform Operation (PO) and Write back (WB). The IF, ID, OF and WB stages take 1 clock cycle each for every instruction. Consider a sequence of 100 instructions. In the PO stage, 40 instructions take 3 clock cycles each, 35 instructions take 2 clock cycles each, and the remaining 25 instructions take 1 clock cycle each. Assume that there are no data hazards and no control hazards.

The number of clock cycles required for completion of execution of the sequence of instructions is _____. **[2018]**

Answer Keys

Exercises

Practice Problems 1

1. A	**2.** D	**3.** B	**4.** B	**5.** B	**6.** D	**7.** C	**8.** D	**9.** C	**10.** D
11. A	**12.** A	**13.** A	**14.** B	**15.** D	**16.** D	**17.** B	**18.** A	**19.** B	**20.** B
21. D									

Practice Problems 2

1. C	**2.** C	**3.** B	**4.** D	**5.** D	**6.** D	**7.** D	**8.** A	**9.** C	**10.** B
11. C	**12.** B	**13.** B	**14.** B	**15.** B	**16.** C	**17.** C	**18.** B	**19.** A	**20.** B

Previous Years' Questions

1. 4	**2.** C	**3.** 1.54	**4.** B	**5.** C	**6.** B	**7.** A	**8.** D	**9.** A	**10.** 3.2
11. 13	**12.** 3	**13** B	**14.** 33.0 : 34.0		**15.** 4	**16.** 1.51	**17.** 219		