- The Sign flag (S) is updated every ADD and SUB instruction.
- The Zero flag (Z) is updated every ADD, SUB, MUL, AND, OR, SLC, and SRC instruction.
- A flag value can only be updated by the instructions related to it.

2.3 Datapath

a) **Stages:** 3

- All instructions regardless of their type must pass through all 3 stages.
- **Instruction Fetch (IF):** Fetches the next instruction from the main memory using the address in the PC (Program Counter), and increments the PC.
- **Instruction Decode (ID):** Decodes the instruction and reads any operands required from the register file.
- **Execute (EX):** Executes the instruction. In fact, all ALU operations are done in this stage. Moreover, it performs any memory access required by the current instruction. For loads, it would load an operand from the main memory, while for stores, it would store an operand into the main memory. Finally, for instructions that have a result (a destination register), it writes this result back to the register file.

b) **Pipeline:** 3 instructions (maximum) running in parallel

- **Number of clock cycles:** $3 + ((n - 1) * 1)$, where n = number of instructions
  - Imagine a program with 7 instructions:
    - $3 + (6 * 1) = 9$ clock cycles
  - You are required to understand the pattern in the example and implement it.

Package 2 Pipeline			
	Instruction Fetch (IF)	**Instruction Decode (ID)**	**Execute (EX)**
Cycle 1	Instruction 1		
Cycle 2	Instruction 2	Instruction 1	
Cycle 3	Instruction 3	Instruction 2	Instruction 1
Cycle 4	Instruction 4	Instruction 3	Instruction 2
Cycle 5	Instruction 5	Instruction 4	Instruction 3
Cycle 6	Instruction 6	Instruction 5	Instruction 4
Cycle 7	Instruction 7	Instruction 6	Instruction 5
Cycle 8		Instruction 7	Instruction 6
Cycle 9			Instruction 7

3 Package 3: Fillet-O-Neumann *with moves on the side*

3.1 Memory Architecture

a) **Architecture:** Von Neumann

- Von Neumann Architecture is a digital computer architecture whose design is based on the concept of stored program computers where **program data** and **instruction data** are stored in the **same memory**.

b) **Memory Size:** 2048 * 32

Main Memory	
2048 Rows	**32 Bits / Row**
	Data (1024 to 2047)
	Instructions (0 to 1023)

- The main memory addresses are from 0 to $2^{11} - 1$ (0 to 2047).
- Each memory block (row) contains 1 word which is 32 bits (4 bytes).
- The main memory is word addressable.
- Addresses from 0 to 1023 contain the program instructions.
- Addresses from 1024 to 2048 contain the data.

c) **Registers:** 33

- Size: 32 bits
- 31 General-Purpose Registers (GPRS)
  - Names: R1 to R31
- 1 Zero Register
  - Name: R0
  - Hard-wired value "0" (cannot be overwritten by any instruction).
- 1 Program Counter
  - Name: PC
  - A program counter is a register in a computer processor that contains the address (location) of the instruction being executed at the current time.
  - As each instruction gets fetched, the program counter is incremented to point to the next instruction to be executed.

3.2 Instruction Set Architecture

a) **Instruction Size:** 32 bits

b) **Instruction Types:** 3

R-Format				
OPCODE	**R1**	**R2**	**R3**	**SHAMT**
4	5	5	5	13

I-Format			
OPCODE	**R1**	**R2**	**IMMEDIATE**
4	5	5	18

J-Format	
OPCODE	**ADDRESS**
4	28

c) **Instruction Count:** 12

- The opcodes are from 0 to 11 according to the instructions order in the following table:

Name	Mnemonic	Type	Format	Operation
Add	ADD	R	ADD R1 R2 R3	R1 = R2 + R3
Subtract	SUB	R	SUB R1 R2 R3	R1 = R2 - R3
Multiply	MUL	R	MUL R1 R2 R3	R1 = R2 * R3
Move Immediate*	MOVI	I	MOVI R1 IMM	R1 = IMM
Jump if Equal	JEQ	I	JEQ R1 R2 IMM	IF(R1 == R2) { PC = PC+1+IMM }
And	AND	R	AND R1 R2 R3	R1 = R2 & R3
Exclusive Or Immediate	XORI	I	XORI R1 R2 IMM	R1 = R2 $\oplus$ IMM
Jump	JMP	J	JMP ADDRESS	PC = PC[31:28] || ADDRESS
Logical Shift Left**	LSL	R	LSL R1 R2 SHAMT	R1 = R2 << SHAMT
Logical Shift Right**	LSR	R	LSR R1 R2 SHAMT	R1 = R2 >>> SHAMT
Move to Register	MOVR	I	MOVR R1 R2 IMM	R1 = MEM[R2 + IMM]
Move to Memory	MOVM	I	MOVM R1 R2 IMM	MEM[R2 + IMM] = R1

* MOVI: R2 will be 0 in the instruction format.
** LSL and LSR: R3 will be 0 in the instruction format.
"||" symbol indicates concatenation (0100 || 1100 = 01001100).

3.3 Datapath

a) **Stages:** 5

- All instructions regardless of their type must pass through all 5 stages even if they do not need to access a particular stage.
- **Instruction Fetch (IF):** Fetches the next instruction from the main memory using the address in the PC (Program Counter), and increments the PC.
- **Instruction Decode (ID):** Decodes the instruction and reads any operands required from the register file.
- **Execute (EX):** Executes the instruction. In fact, all ALU operations are done in this stage.
- **Memory (MEM):** Performs any memory access required by the current instruction. For loads, it would load an operand from the main memory, while for stores, it would store an operand into the main memory.
- **Write Back (WB):** For instructions that have a result (a destination register), the Write Back writes this result back to the register file.

b) **Pipeline:** 4 instructions (maximum) running in parallel

- **Instruction Fetch (IF)** and **Memory (MEM)** can not be done in parallel since they access the same physical memory.
- At a given clock cycle, you can either have the **IF, ID, EX, WB** stages active, or the **ID, EX, MEM, WB** stages active.
- **Number of clock cycles:** $7 + ((n - 1) * 2)$, where n = number of instructions
  - Imagine a program with 7 instructions:
    - $7 + (6 * 2) = 19$ clock cycles
  - You are required to understand the pattern in the example and implement it.

Package 3 Pipeline					
	Instruction Fetch (IF)	**Instruction Decode (ID)**	**Execute (EX)**	**Memory (MEM)**	**Write Back (WB)**
Cycle 1	Instruction 1				
Cycle 2		Instruction 1			
Cycle 3	Instruction 2	Instruction 1			
Cycle 4		Instruction 2	Instruction 1		
Cycle 5	Instruction 3	Instruction 2	Instruction 1		
Cycle 6		Instruction 3	Instruction 2	Instruction 1	
Cycle 7	Instruction 4	Instruction 3	Instruction 2		Instruction 1
Cycle 8		Instruction 4	Instruction 3	Instruction 2	
Cycle 9	Instruction 5	Instruction 4	Instruction 3		Instruction 2
Cycle 10		Instruction 5	Instruction 4	Instruction 3	
Cycle 11	Instruction 6	Instruction 5	Instruction 4		Instruction 3
Cycle 12		Instruction 6	Instruction 5	Instruction 4	
Cycle 13	Instruction 7	Instruction 6	Instruction 5		Instruction 4
Cycle 14		Instruction 7	Instruction 6	Instruction 5	
Cycle 15		Instruction 7	Instruction 6		Instruction 5
Cycle 16			Instruction 7	Instruction 6	
Cycle 17			Instruction 7		Instruction 6
Cycle 18				Instruction 7	
Cycle 19					Instruction 7

- The pattern is as follows:
  - You fetch an instruction every 2 clock cycles starting from clock cycle 1.
  - An instruction stays in the Decode (ID) stage for 2 clock cycles.
  - An instruction stays in the Execute (EX) stage for 2 clock cycles.
  - An instruction stays in the Memory (MEM) stage for 1 clock cycle.
  - An instruction stays in the Write Back (WB) stage for 1 clock cycle.
  - You can not have the Instruction Fetch (IF) and Memory (MEM) stages working in parallel. Only one of them is active at a given clock cycle.

4 Package 4: Double Big Harvard *combo large arithmetic shifts*

4.1 Memory Architecture

a) **Architecture:** Harvard

- Harvard Architecture is the digital computer architecture whose design is based on the concept where there are **separate storage** and **separate buses (signal path)** for **instruction** and **data**. It was basically developed to overcome the bottleneck of Von Neumann Architecture.

b) **Instruction Memory Size:** 1024 * 16

Instruction Memory	
	16 Bits / Row
1024 Rows	

- The instruction memory addresses are from 0 to $2^{10} - 1$ (0 to 1023).
- Each memory block (row) contains 1 word which is 16 bits (2 bytes).