Computer Architecture

Phase 1

Name	SEC	BN
Ahmed Mohamed Soliman	1	9
Hassan Osama	1	16
Hussein Youssef	1	17
Muhammed Abdul Kariim	2	20

The instructions in our project are classified as below:

- ALU instructions
  - Two-op
    - Immediate instructions.
    - Non-immediate instructions.
  - Not Two-op
- Not ALU instructions
  - 1st Block.98
  - 2nd Block, Memory access.
  - 3rd Block, Jumps.

ALU

Two-op

Immediate Instructions

ALU	2Operand	Immediate	Opcode		Shift		Immediate Value							$Register_{src}$ Index		
1	1	1	-		0	1	-	-	-	-	-	-	-	-	-	-

Op-Codes

Instruction	Op-code
SHR R_{dst}, Immediate	**0**
SHL R_{dst}, Immediate	**1**

Non-Immediate Instructions

ALU	2Operand	Immediate	Opcode		Add/Sub	Don't Care Values				$Register_{dst}$ Index			$Register_{src}$ Index		
1	1	0	-	-	0	X	X	X	X	-	-	-	-	-	-

Op-Codes

Instruction	Op-code
ADD R_{src}, R_{dst}	**00**
SUB R_{src}, R_{dst}	**01**
AND R_{src}, R_{dst}	**10**
OR R_{src}, R_{dst}	**11**

Non-Two-op

ALU	2Operand	Don't Care	Opcode			Don't Care Values							$Register_{dst}$ Index		
1	0	X	-	-	-	X	X	X	X	X	X	X	-	-	-

Op-codes

Instruction	Op-code
Not R_{dst}	**000**
INC R_{dst}	**001**
DEC R_{dst}	**010**
PUSH R_{dst}	**011**
POP R_{dst}	**100**
CALL R_{dst}	**101**
RET	**110**
RTI	**111**

The remaining instructions are classified as Not ALU instructions. These instructions are distributed to 3 different blocks.

Not ALU

1st Block

ALU	Block		Opcode		Don't Care					$Register_{dst}$ Index			$Register_{src}$ Index		
0	0	0	-	-	X	X	X	X	X	-	-	-	-	-	-

Op-Codes

Instruction	Op-code
NOP	**00**
MOV R_{src}, R_{dst}	**01**
CLRC	**10**
SETC	**11**

2nd Block – Memory Access

ALU	Block		Opcode			Don't care				$Register_{dst}$ Index			$Register_{src}$ Index		
0	0	1	-	-	-	X	X	X	X	-	-	-	-	-	-

- Note: This is a dynamic IR, as these sections varies from the instruction to the other. Ex: in LDM instruction, the section of R_{src} will be merged with the immediate value section. Another example, in LDD instruction, there will not be immediate value section. And so on.

Op-Codes

Instruction	Op-code
IN R_{dst}	**000**
OUT R_{dst}	**001**
LDM R_{dst}, Immediate	**010**
LDD R_{src}, R_{dst}	**011**
STD R_{src}, R_{dst}	**100**

3rd Block – Jumps

ALU	Block		Opcode		Don't care values								Register$_{dst}$ Index		
0	1	0	-	-	X	X	X	X	X	X	X	X	-	-	-

Op-Codes

Instruction	Op-code
JZ R_{dst}	**00**
JN R_{dst}	**01**
JC R_{dst}	**10**
JMP R_{dst}	**11**

Control Signals

Label	Bits	Label	Bits
ALU Op-Code	3	SP	32
Op1	16	2-op	1
Op2	16	1OR2	1
PC	32	MEM-OR-ALU	1
POP	1	PUSH	1
MEMwr	1	MEMrd	1
Address	16	R_{dst} idx	3
R_{src} idx	3	IMM	1
WB	1	EN-SP	1

Stall & Flush

Stall:

FD = the buffer between Fetch and Decode.

DE = the buffer between Decode and Execute

- **Load/Pop Use** 1 stall

Stall = **[**(ALU AND not (RET AND RTI AND POP)) AND (

((DE1.rd AND DE1.Rdst == FD1.Rdst)) OR (2Op1 AND ((DE1.rd AND DE1.Rdst == FD1.Rsrc))) OR

((DE2.rd AND DE2.Rdst == FD1.Rdst)) OR (2Op1 AND ((DE2.rd AND DE2.Rdst == FD1.Rsrc))) OR

((DE1.rd AND DE1.Rdst == FD2.Rdst)) OR (2Op2 AND ((DE1.rd AND DE1.Rdst == FD2.Rsrc))) OR

((DE2.rd AND DE2.Rdst == FD2.Rdst)) OR (2Op2 AND ((DE2.rd AND DE2.Rdst == FD2.Rsrc))) OR

(FD1.Rdst == FD2.Rdst OR (2Op2 AND FD1.Rdst == FD2.Rsrc)) OR

(2Op2 AND FD1.rd AND DE2.Rsrc == FD1.Rdst))**]**

OR [((MOV OR LDD OR STD) AND (FD1.Rdst == FD2.Rsrc)) OR

((MOV OR LDD OR STD) AND (DE1.rd AND DE1.Rdst == FD2.Rsrc)) OR

(STD OR OUT OR any JMP) AND (FD1.Rdst == FD1.Rdst) OR

(STD OR OUT OR any JMP) AND (DE1.rd AND DE1.Rdst == FD2.Rdst)**]**

- Disable IR & PC (to keep the instruction)
- Reset FD1 or FD2/DE1 or DE2.

Flush

Taken = (Branch-Taken OR JMP OR CALL OR CALL OR RET OR RTI).

When to flush?

If Taken then:

- If we are in the first channel, then we should clear IR of the first and second channel and rise the No-Op bit for the instructions in the previous stage and for the current instruction in the second channel should be flushed.
- But if we are in the second stage, same as previous will be done except the part of flushing the instruction of the second channel.

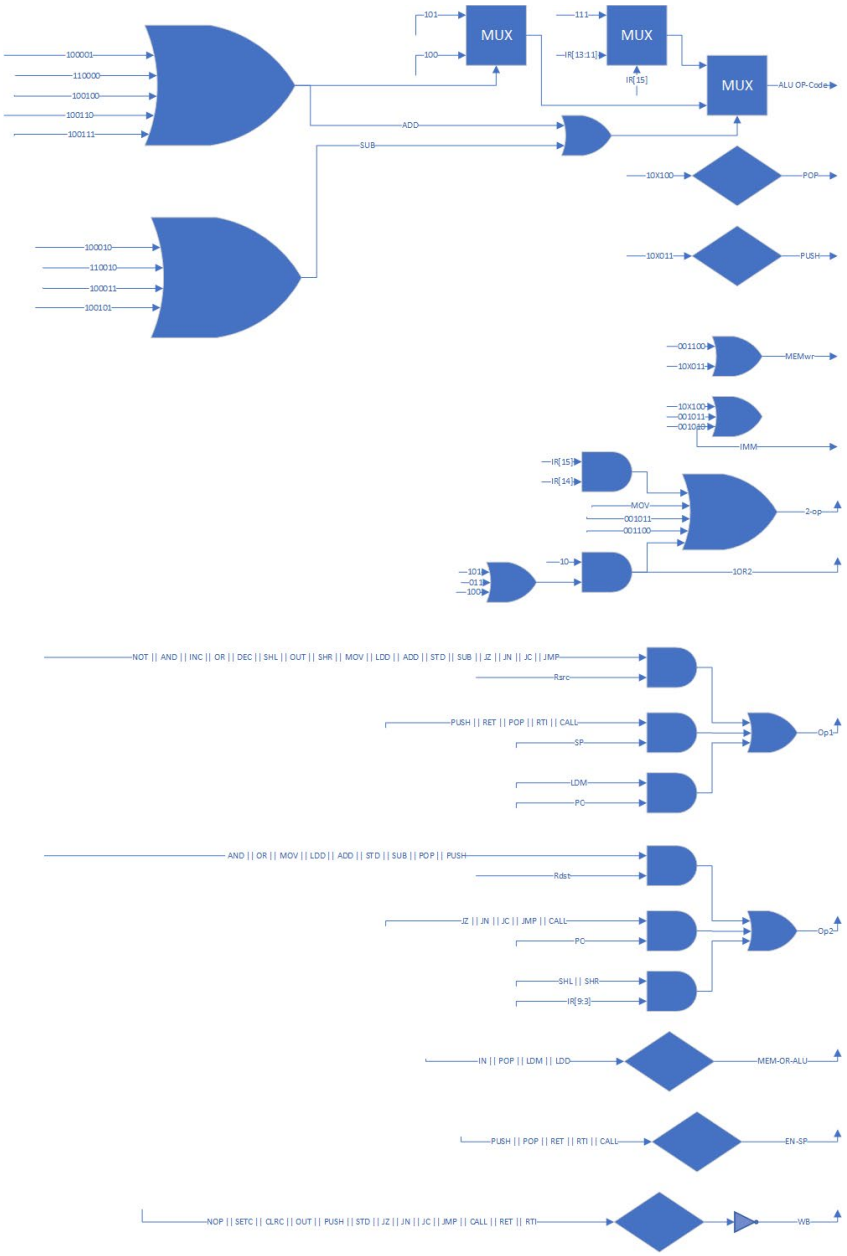
Data Forwarding

- Rdst from EM when: [DE.Rdst == EM.Rdst AND not(MOV || LDM)]
- Rsrc from EM when: [DE.Rsrc == EM.Rdst AND 2-op]

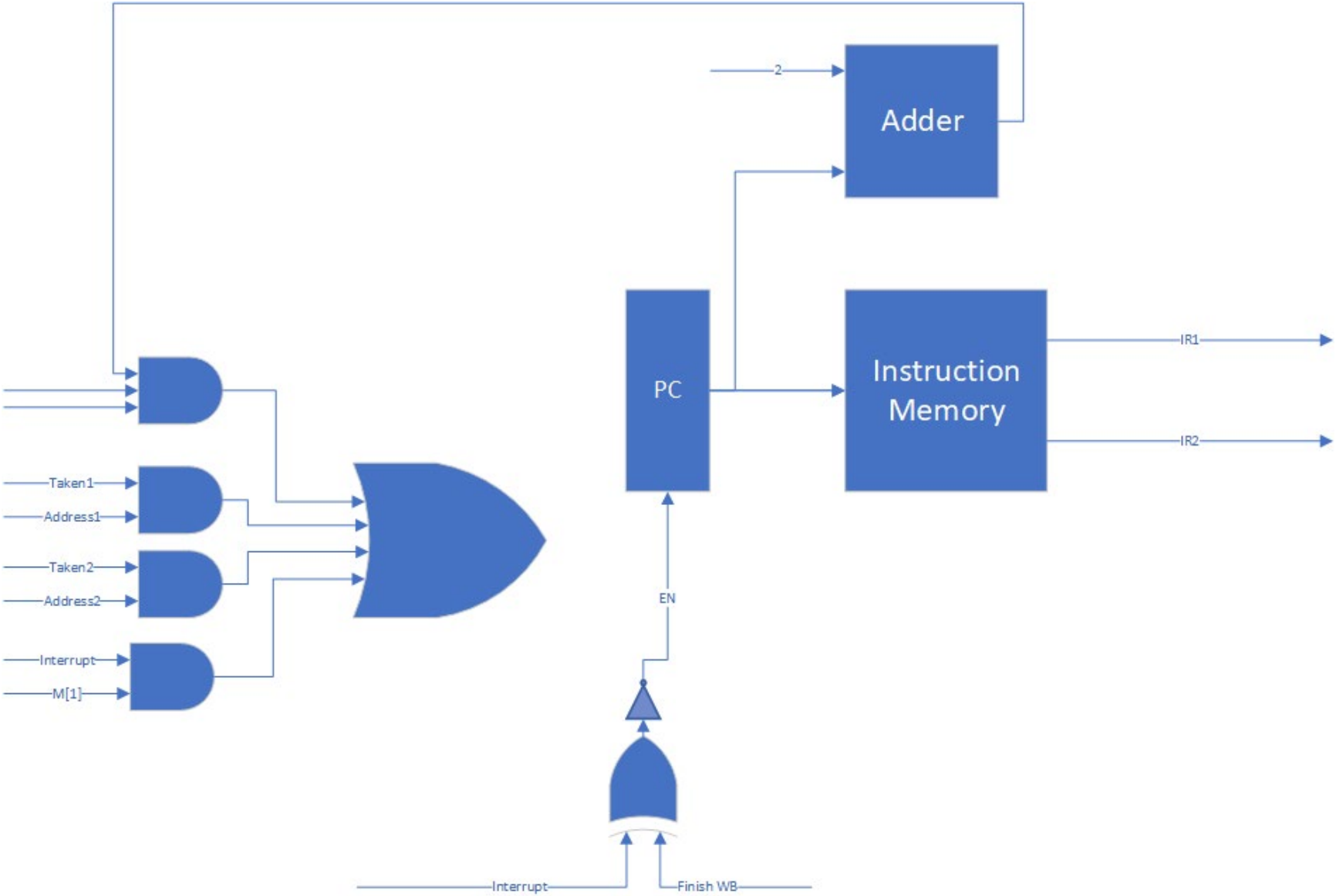
Same for MW.

Diagrams

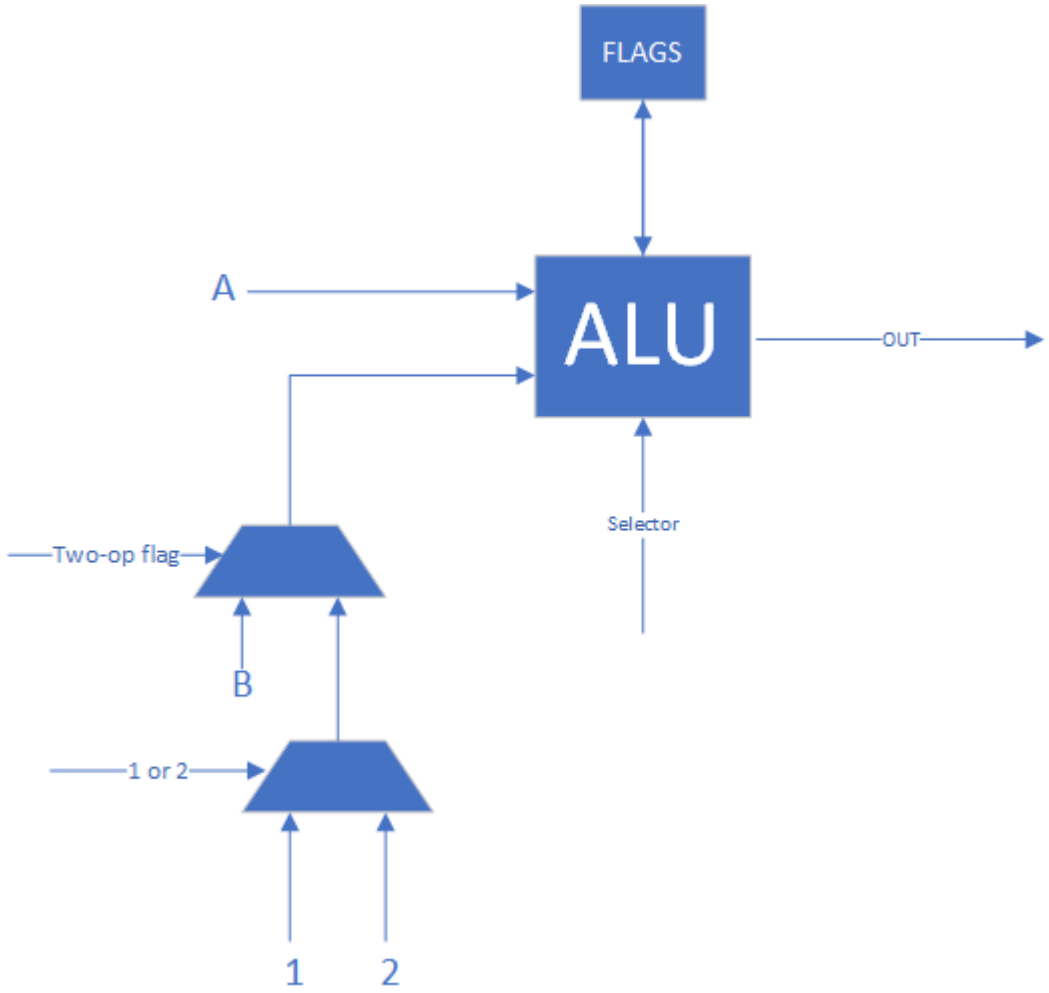
Control Unit



Fetch


2
Adder
PC
Instruction Memory
IR1
IR2
Taken1
Address1
Taken2
Address2
Interrupt
M[1]
EN
Interrupt
Finish WB

Execute



Write Back

MUX states:

EN1	EN2	OUT
0	0	SP
0	1	Data1
1	0	Data2
1	1	ERROR