Cairo university

Faculty of Engineering

Architecture Project

Phase 1

Submitted to:

Eng. Maheed Hatem

Submitted by:

Ahmed Magdy	Sec: 1 BN: 5
Mohamed Ibrahim	Sec: 2 BN: 12
Omar Tarek	Sec: 2 BN: 7
Mohamed Abobakr	Sec: 2 BN: 13

Instruction format:

32-bit Instruction divided into 4 types:

1) One - operand
2) Two - operand
3) Memory
4) Branching

1) One- operand

2-bits(0 0)	4-bits	3-bits	23-bit
Instruction type	Functions	Rdst	don't-care(X)

Function	Op-code
NOP	0000
SETC	0001
CLRC	0010
CLR Rdst	0011
NOT Rdst	0100
INC Rdst	0101
DEC Rdst	0110
NEG Rdst	0111
OUT Rdst	1000
IN Rdst	1001
RLC Rdst	1010
RRC Rdst	1011

2) Two- Operand

2-bits(0 1)	3-bits	3-bits	3-bits	5-bits	16-bits
Instruction type	Operation	Rsrc	Rdst	Shamd	Immediate value

Function	OPcode
MOV Rsrc, Rdst	000
ADD Rsrc, Rdst	001
SUB Rsrc, Rdst	010
AND Rsrc, Rdst	011
OR Rsrc, Rdst	100
IADD Rdst, Imm	101
SHL Rsrc, Imm	110
SHR Rsrc, Imm	111

3) Memory

2-bits(1 0)	3-bits	3-bits	3-bits	5-bits	16-bits
Instruction type	Operation	Rsrc	Rdst	Don't care (X)	Immediate value / Offset

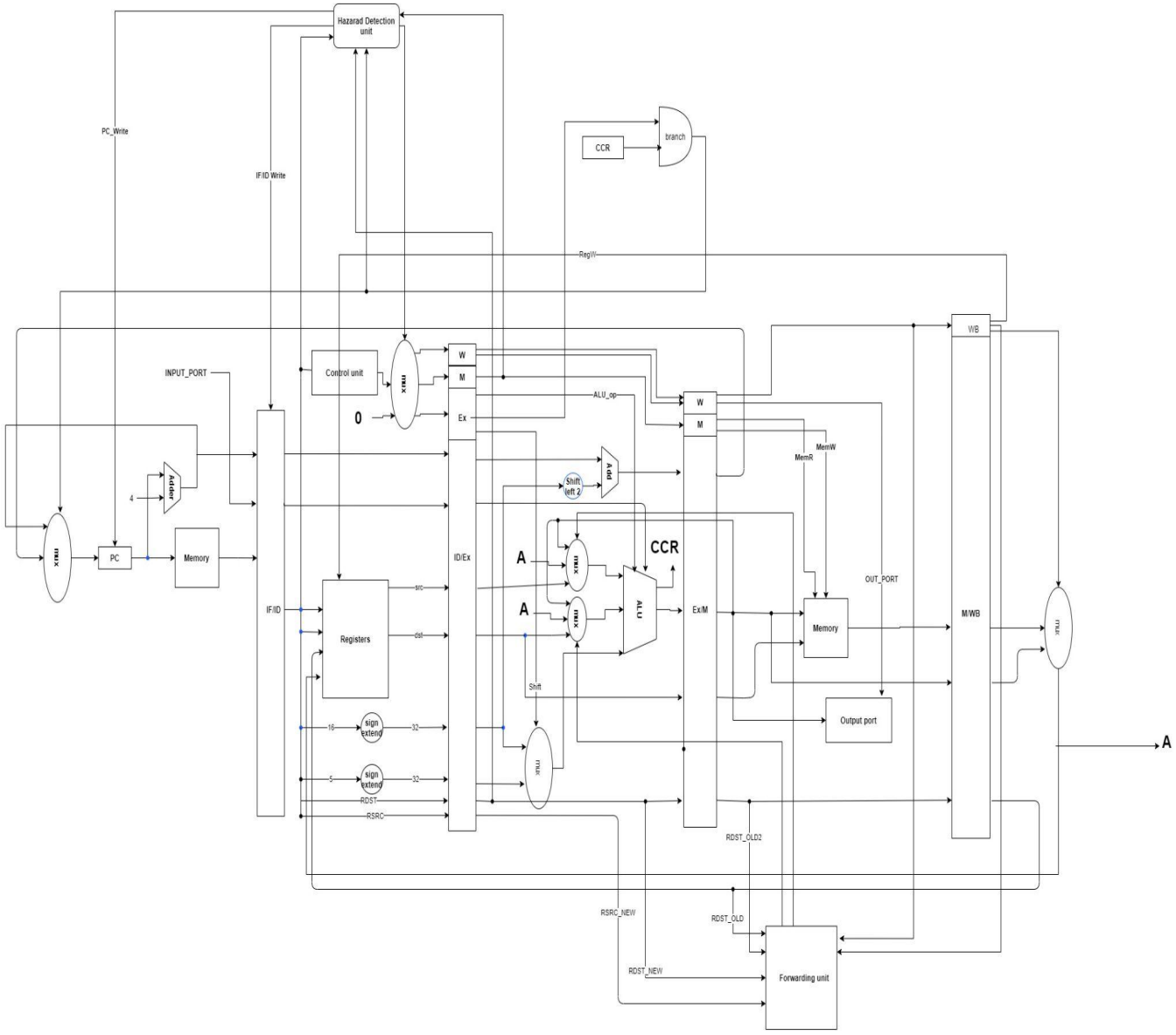
Function	OPcode
PUSH Rdst	000
POP Rdst	001
LDM Rdst, Imm	010
LDD Rdst, offset(Rsrc)	011
STD Rdst, offset(Rsrc)	100

4)Branching

2-bits(1 1)	3-bits	3-bits	24-bit
Instruction type	Operation	Rdst	Don't care

Function	OPcode
JZ Rdst	000
JN Rdst	001
JC Rdst	010
JMP Rdst	011
Call Rdst	100
RET	101
RTI	110

Schematic Diagram



Control Signals

Signals:

1) ALU_Op
2) Immediate(0)/ shift(1)
3) Register Write
4) Memory Read
5) Memory Write
6) Branch
7) Memory to Register(0 from reg to reg, 1 from memory to reg)
8) Out_port

Instruction	ALU_op	Imm / Shift	Branch	Memory Read	Memory Write	Register Write	MemoryTo Register	Out_port
NOP	00 + 0000	X	0	0	0	0	X	0
SETC	00 + 0001	X	0	0	0	0	X	0
CLRC	00 + 0010	X	0	0	0	0	X	0
CLR	-- edited 00 + 1111	X	0	0	0	1	0	0
NOT	00 + 0100	X	0	0	0	1	0	0
INC	00 + 0101	X	0	0	0	1	0	0
DEC	00 + 0110	X	0	0	0	1	0	0
NEG	00 + 0111	X	0	0	0	1	0	0
OUT	00 + 1000	X	0	0	0	0	X	1
IN	--edited 00 + 1101	X	0	0	0	1	X	0
MOV	01 + 000X	X	0	0	0	1	0	0
ADD	01 + 001X	X	0	0	0	1	0	0
SUB	01 + 010X	X	0	0	0	1	0	0
AND	01 + 011X	X	0	0	0	1	0	0

OR	01 + 100X	X	0	0	0	1	0	0
IADD	01 + 101X	0	0	0	0	1	0	0
SHL	01 + 110X	1	0	0	0	1	0	0
SHR	01 + 111X	1	0	0	0	1	0	0
RLC	00 + 1010	X	0	0	0	1	0	0
RRC	00 + 1011	X	0	0	0	1	0	0
PUSH	10 + 000X	X	0	0	1	0	0	0
POP	10 + 001X	X	0	1	0	1	1	0
LDM	10 + 010X	0	0	0	0	1	0	0
LDD	10 + 011X	0	0	1	0	1	1	0
STD	10 + 100X	0	0	0	1	0	0	0
JZ	11 + 000X	X	1	0	0	0	X	0
JN	11 + 001X	X	1	0	0	0	X	0
JC	11 + 010X	X	1	0	0	0	X	0
JMP	11 + 011X	X	1	0	0	0	X	0
CALL	11 + 100X	X	1	0	0	0	0	0
RET	11 + 101X	X	1	1	0	0	0	0
RTI	11 + 110X	X	1	1	0	0	0	0

Pipeline Stages Design:

Pipeline Registers Details:

Register	Size in bits	Inputs
IF/ID	80	[15 : 0] Instruction [47-16] : Incremented PC [79-48] : IN.value
ID/EX	211	[31:0]: Incremented PC(forwarded) [63:32]: R[Rsrc] [95:64]: R[Rdst] [127:96]: Immediate Value [159:128]: Shift Amount [162:160]: Rsrc [165:163]: Rdst [197:166]: IN value (forwarded) [210:198]: Control Signals
EX/M	107	[31:0]: ALU Result [34:32]: CCR [66:35]: PC with branching [98:67]: R[Rdst](forwarded) [101:99]: Rdst(forwarded) [106:102]: Memory and WB Control Signals(forwarded)
M/WB	69	[31:0]: M[ALUresult] [63:32]: ALUresult(forwarded) [66:64]: Rdst(forwarded) [68:67]: WB control signals

Pipeline Hazard:

1) **Structural Hazard: No Structural Hazard as we are implementing Harvard Architecture so the data and instruction memories are separated, and we Use the first half of the cycle in write and the other half in read.**

2) **Data Hazard: There is when if instruction has a write back (RegToReg) and there are 2 instructions after it and have Rdst equals Rsrc or Rdst of the first instruction, if the write back is MemoryToReg, then the hazard is in the next instruction only.**

3) **Control Hazard: control hazard happens when there is a branch instruction in the pipe and the prediction was incorrect. In this case we flush the instructions in the fetch and the decode stages and fetch the correct address. This functionality is done using the hazard detection unit.**

Solutions:

1) **Data Forwarding: Solves all data Hazards except Load-Use which is detected by the Hazard detection unit and we Stall.**

2) **Hazard Detection Unit: Detects Data Hazards and stall.**

3) **Static Branch Prediction : the hazard detection unit flushes the instructions by zeroing the opcode in the first buffer IF/ID and setting the mux to zero the control unit output.**