Cairo University Faculty of Engineering
Computer Engineering Department

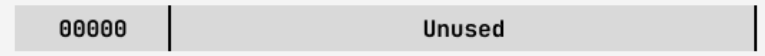
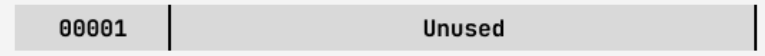
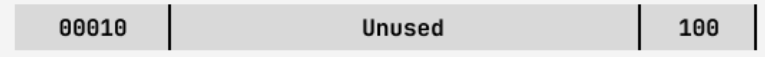
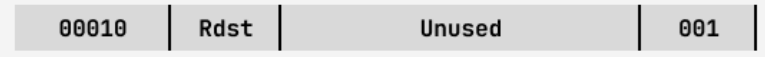
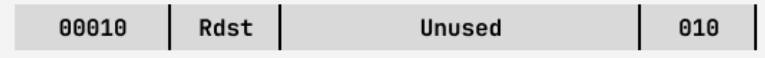
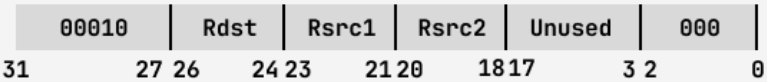
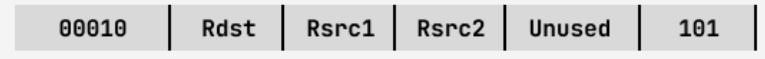
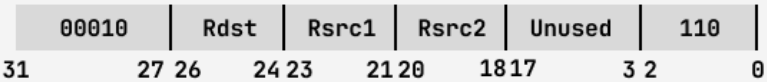
PHASE 1 COMPUTER ARCHITECTURE PROJECT

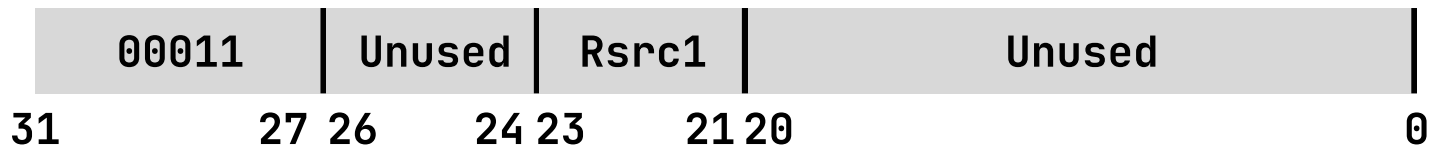
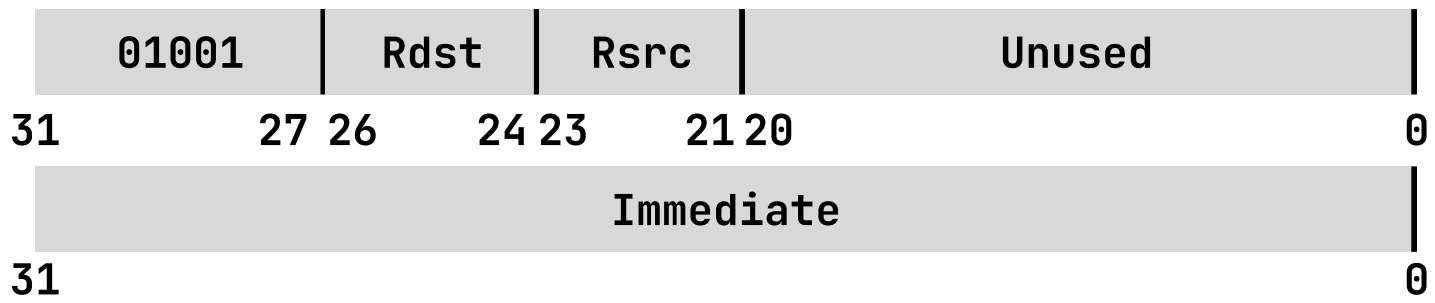
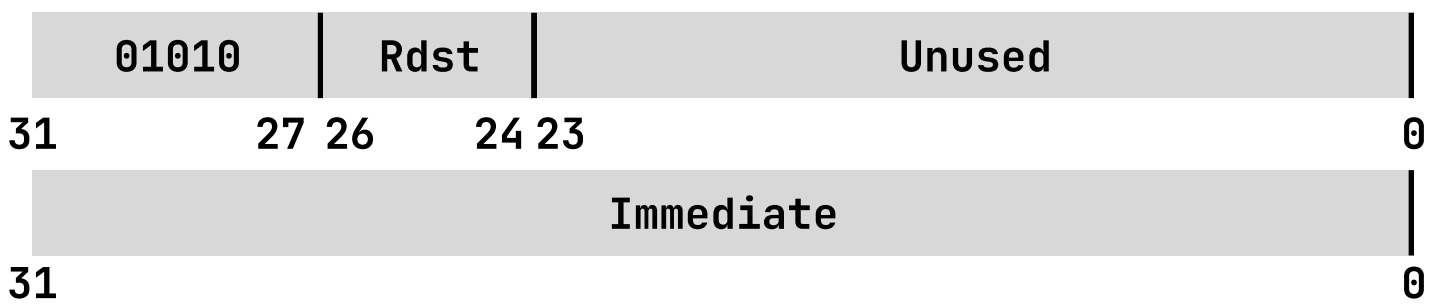
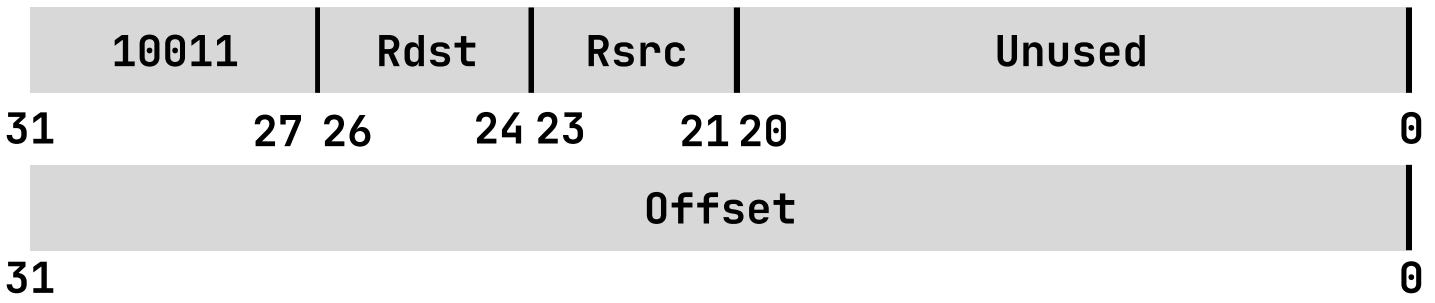
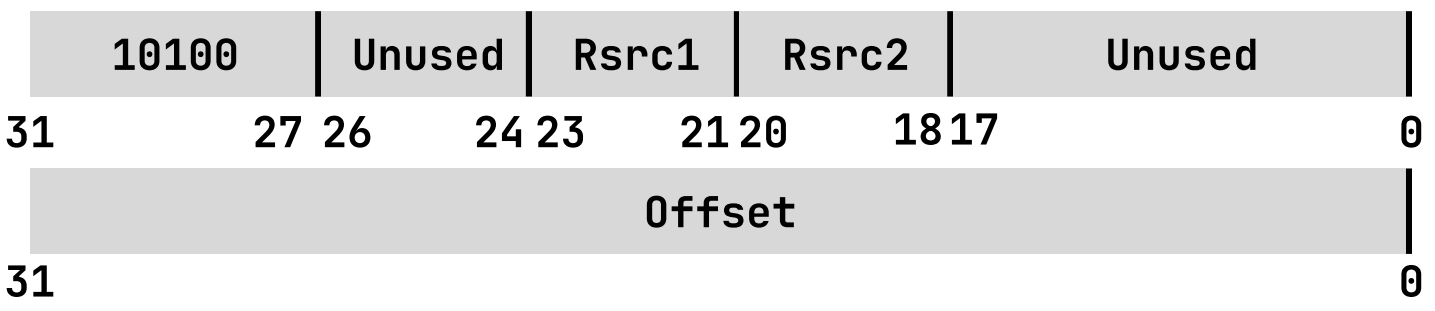
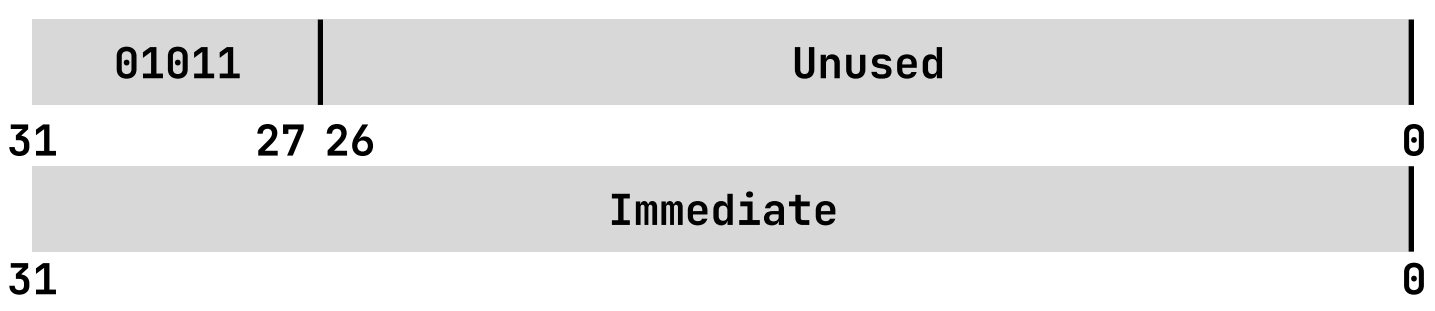
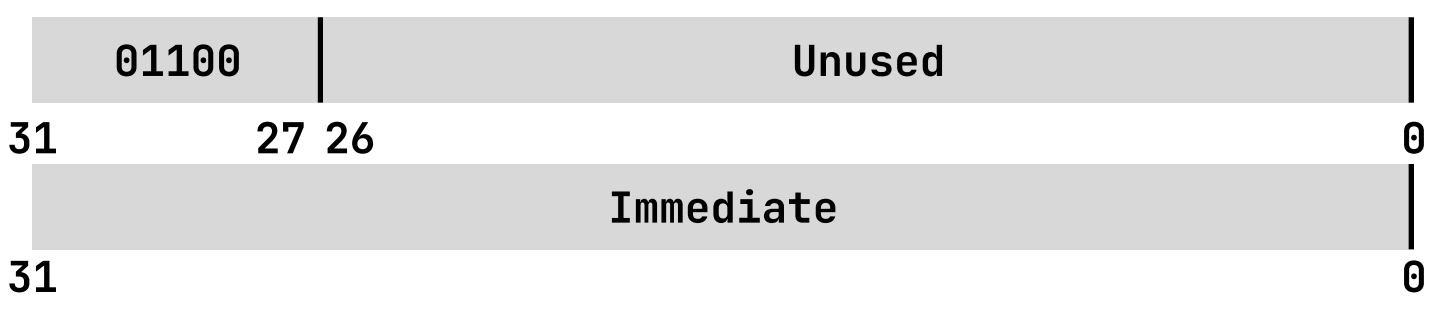
Name	Code
Abdelrahman Wissam Abdelrahman Elbana	91240450
Ahmed Mahmoud Kamal	91240129
Seif Atef Mousa Mohamed	91240378
Seif Hesham AbdelMonem	91240379

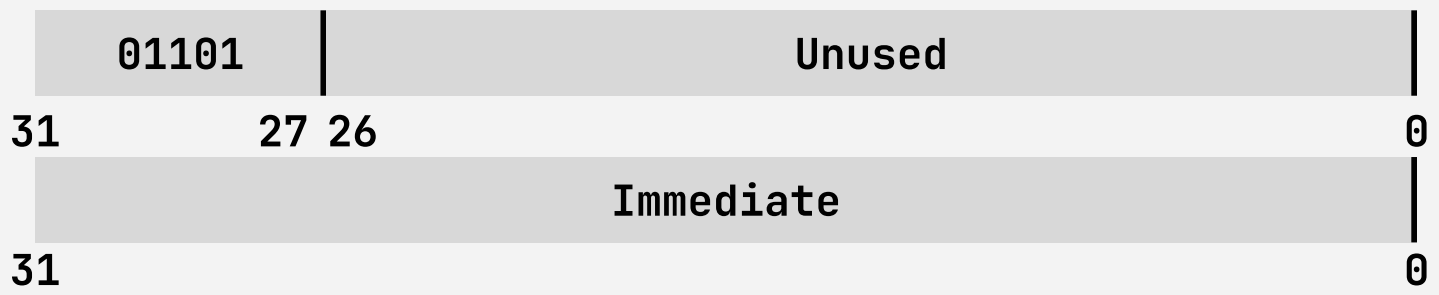
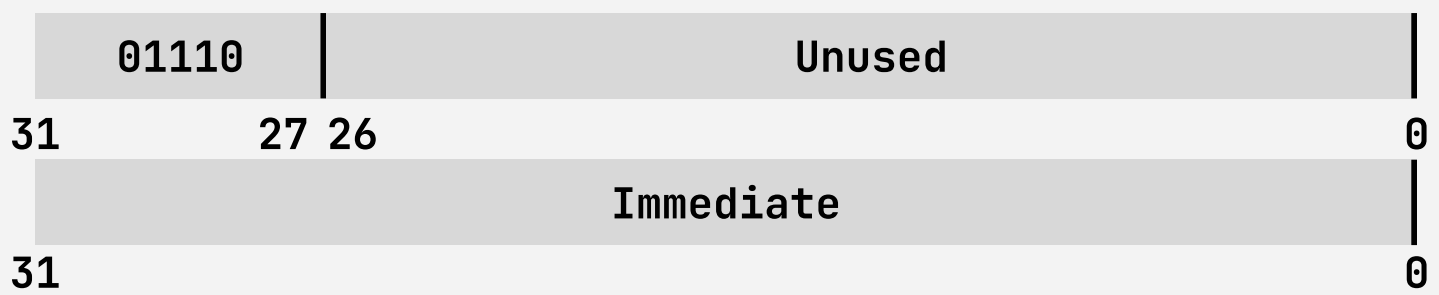
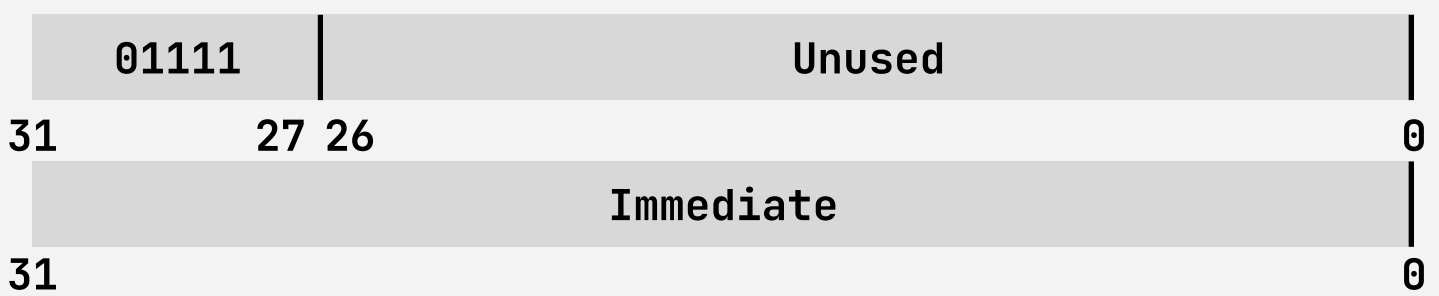
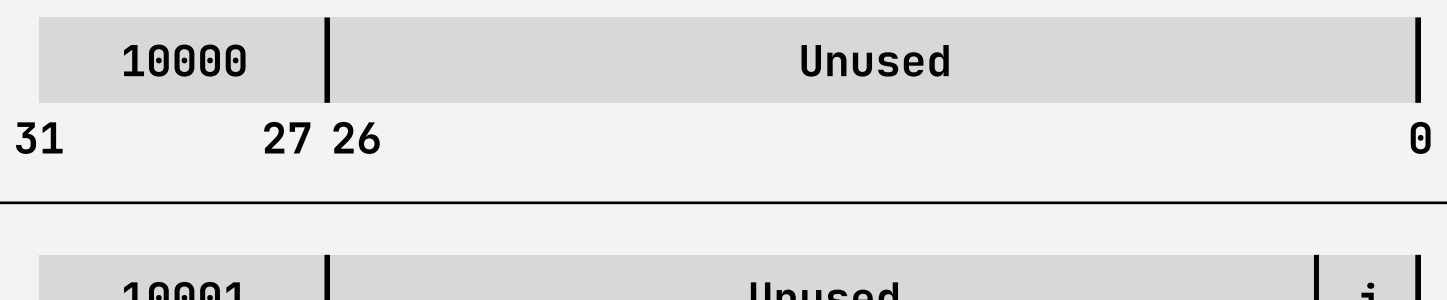
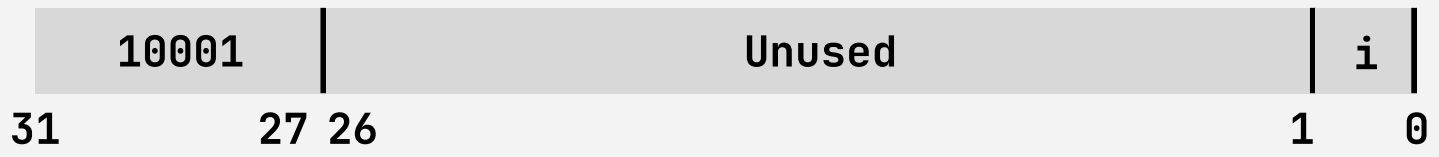
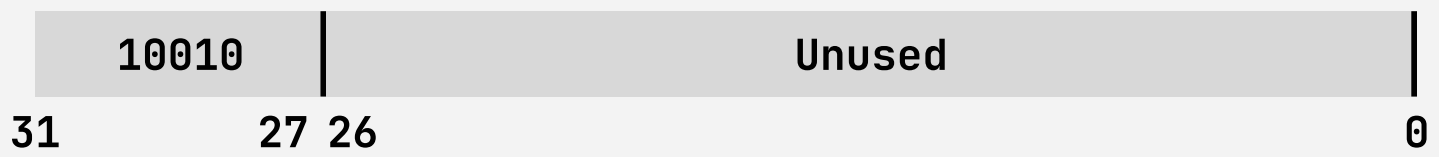
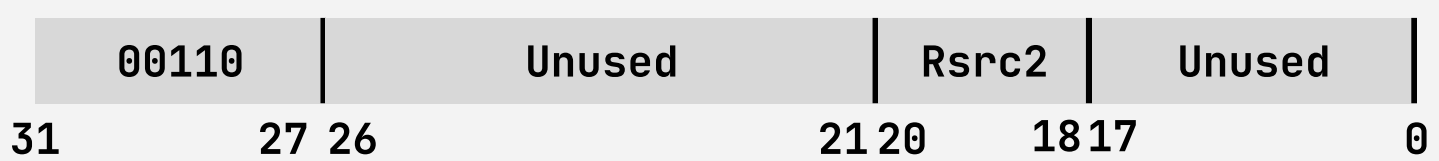
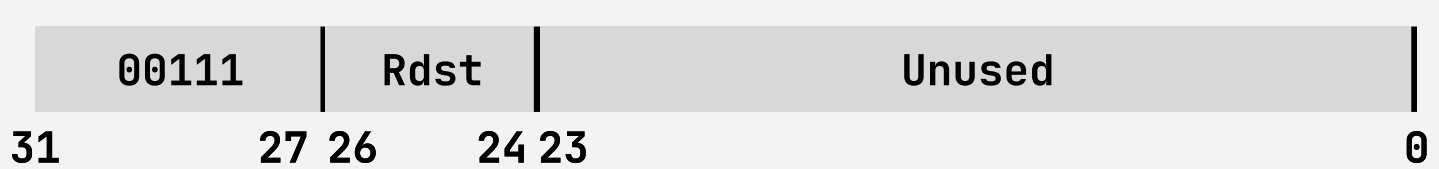
NOVEMBER 30, 2025

Computer Engineering Cairo University

Instruction Bits

Instruction	Instruction Bits
NOP	
HLT	
SETC	
NOT Rdst	
INC Rdst	
MOV Rsrc,Rdst	
MOVS Rsrc,Rdst	
SWAP Rsrc,Rdst	
ADD Rdst, Rsrc1, Rsrc2	
SUB Rdst, Rsrc1, Rsrc2	
AND Rdst, Rsrc1, Rsrc2	

Instruction	Instruction Bits
OUT Rsrc1	
IN Rdst	
IADD Rdst, Rsrc, Imm	
LDM Rdst, Imm	
LDD Rdst, offset(Rsrc)	
STD Rsrc2, offset(Rsrc1)	
JZ Imm	
JN Imm	

Instruction	Instruction Bits
JC Imm	
JMP Imm	
CALL Imm	
RET	
INT index	
RTI	
PUSH Rsrc2	
POP Rdst	

Pipeline Register Details

IF/ID Register:

- Immediate = 32 bits
- PC = 32 bits
- Instruction = 32 bits
- Total = 96 bits

ID/EX Register:

- WB = 2 bits *(Control Signals)*
- M = 2 bits *(Control Signals)*
- EX = 8 bits *(Control Signals)*
- PC = 32 bits
- ReadData1 = 32 bits
- ReadData2 = 32 bits
- Immediate/Offset = 32 bits
- Rsrc1 = 3 bits
- Rsrc2 = 3 bits
- Rdst = 3 bits
- Funct = 3 bits
- Total = 159 bits

EX/MEM Register:

- PC = 32 bits
- ALUResult = 32 bits
- StoreData = 32 bits
- CCR/Flags = 3 bits
- WB = 2 bits *(Control Signals)*
- M = 9 bits *(Control Signals)*
- Total = 113 bits

MEM/WB Register:

- WriteAddr = 3 bits
- MEMResult = 32 bits
- ALUResult = 32 bits
- WB = 3 bits *(Control Signals)*
- Total = 70 bits

Pipeline Hazards

Static Branch Prediction

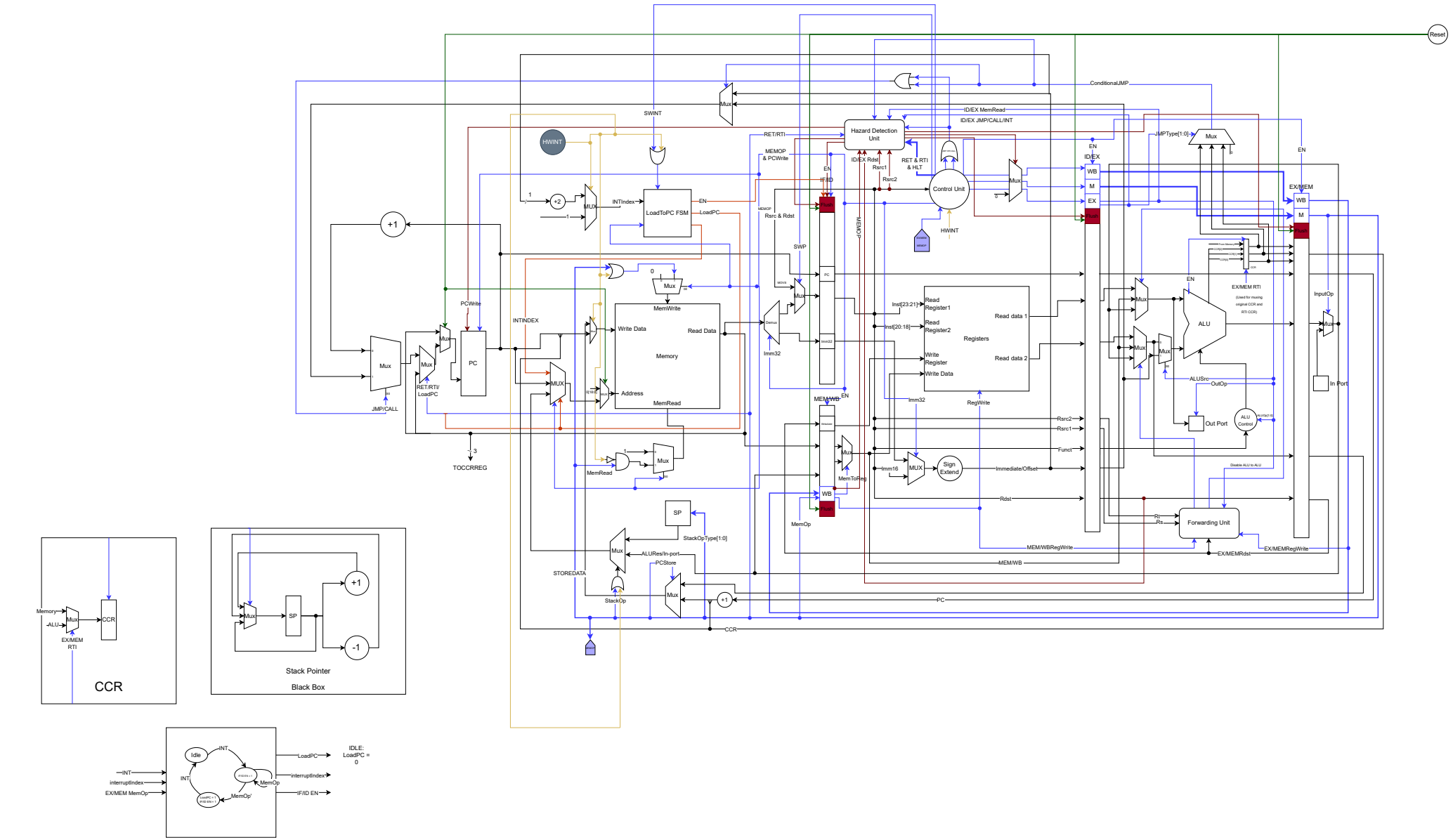
- Static branch prediction is not taken

Hazards

Hazard Type	Solution
Memory Hazard	Memory to ALU forwarding
Execution Hazard	ALU to ALU forwarding
Load Use Hazard	Stall
Conditional Jump followed by Unconditional Jump	Priority to Conditional Jump & flush younger instructions
Unconditional Jump & Call & Interrupt	Flushing younger instructions after instructions reaches execute stage
Interrupt needs to store PC to the stack & load Interrupt vector table address to PC	Use a separate FSM to handle loading the Interrupt table Address to the PC whenever memory is free
Fetch and memory operation at the same time	Send EX/MEM MEMOp signal to control unit to stop fetch and prioritize memory operation. MEM/WB MEMOp is passed to HDU to flush duplicated instruction in the IF/ID Register that resulted from stalling.
SWAP Instruction	Split into 2 MOV instructions with the second one being a special MOVS that disables ALU to ALU forwarding
32-bit Immediate	Immediate is loaded over two cycles while stalling the pipeline
OutPort forwarding	Placing OutPort in EX stage and taking Rsrc1 after the forwarding Mux
Unconditional Jump decision	Taken in EX stage to have access to latest CCR/Flags
InPort forwarding	Place right after EX/MEM Register to allow ALU to ALU forwarding
Conditional Jump followed by HLT	No stalling caused by HLT as Hazard detection unit gives priority to Unconditional Jump
RET & RTI	IF/ID is stalled by HDU until EX/MEM RET/RTI signal is received by HDU, Pipeline is flushed

Schematic Design

Instruction	Control Signals																							
/	ALUSrc	OutOp	InOp	MemtoReg	RegWrite	RET	RTI	StackOpType[1:0]	PCStore	MemOp	MemRead	MemWrite	SWINT	JMP/Call	Imm32	SWP	PCWrite	IF/ID EN	ID/EX EN	EX/MEM EN	MEM/WB EN	ALUOp[1:0]	JMPType[1:0]	MOVS	HLT
NOP	0	0	0	0	0	0	0	00	0	0	0	0	0	0	0	0	0	0	0	0	0	00	00	0	0
HLT	0	0	0	0	0	0	0	00	0	0	0	0	0	0	0	0	1	0	0	0	0	00	00	0	1
SETC	0	0	0	0	0	0	0	00	0	0	0	0	0	0	0	0	0	0	0	0	0	11	00	0	0
NOT Rdst	0	0	0	0	1	0	0	00	0	0	0	0	0	0	0	0	0	0	0	0	0	11	00	0	0
INC Rdst	0	0	0	0	1	0	0	00	0	0	0	0	0	0	0	0	0	0	0	0	0	11	00	0	0
OUT Rsrc1	0	1	0	0	0	0	0	00	0	0	0	0	0	0	0	0	0	0	0	0	0	00	00	0	0
IN Rdst	0	0	1	0	1	0	0	00	0	0	0	0	0	0	0	0	0	0	0	0	0	00	00	0	0
MOV Rsrc, Rdst	0	0	0	0	1	0	0	00	0	0	0	0	0	0	0	0	0	0	0	0	0	11	00	0	0
SWAP Rsrc, Rdst	0	0	0	0	1	0	0	00	0	0	0	0	0	0	0	1	0	0	0	0	0	11	00	0	0
ADD Rdst, Rsrc1, Rsrc2	0	0	0	0	1	0	0	00	0	0	0	0	0	0	0	0	0	0	0	0	0	11	00	0	0
SUB Rdst,Rsrc1, Rsrc2	0	0	0	0	1	0	0	00	0	0	0	0	0	0	0	0	0	0	0	0	0	11	00	0	0
AND Rdst, Rsrc1, Rsrc2	0	0	0	0	1	0	0	00	0	0	0	0	0	0	0	0	0	0	0	0	0	11	00	0	0
IADD Rdst, Rsrc,Imm	1	0	0	0	1	0	0	00	0	0	0	0	0	0	1	0	1	0	1	1	1	11	00	0	0
PUSH Rsrc	0	0	0	0	0	0	0	10	0	1	0	1	0	0	0	0	0	0	0	0	0	01	00	0	0
POP Rdst	0	0	0	1	1	0	0	11	0	0	1	0	0	0	0	0	0	0	0	0	0	00	00	0	0
LDM Rdst, Imm	1	0	0	0	1	0	0	00	0	0	0	0	0	0	1	0	1	0	1	1	1	01	00	0	0
LDD Rdst, offset(Rsrc1)	1	0	0	1	1	0	0	00	0	1	1	0	0	0	1	0	1	0	1	1	1	11	00	0	0
STD Rsrc2. offset(Rsrc1)	1	0	0	0	0	0	0	00	0	1	0	1	0	0	1	0	1	0	1	1	1	11	00	0	0
JZ Imm	0	0	0	0	0	0	0	00	0	0	0	0	0	0	1	0	1	0	1	1	1	00	01	0	0
JN Imm	0	0	0	0	0	0	0	00	0	0	0	0	0	0	1	0	1	0	1	1	1	00	10	0	0
JC Imm	0	0	0	0	0	0	0	00	0	0	0	0	0	0	1	0	1	0	1	1	1	00	11	0	0
JMP Imm	0	0	0	0	0	0	0	00	0	0	0	0	0	1	1	0	1	1	1	1	1	00	00	0	0
CALL Imm	0	0	0	0	0	0	0	10	1	1	0	1	0	1	1	0	1	1	1	1	1	01	00	0	0
RET	0	0	0	0	0	1	0	11	0	1	1	0	0	0	0	0	0	0	0	0	0	00	00	0	0
INT index	0	0	0	0	0	0	0	10	1	0	0	0	1	0	0	0	0	0	0	0	0	00	00	0	0
RTI	0	0	0	0	0	1	1	11	0	1	1	0	0	0	0	0	0	0	0	0	0	00	00	0	0
MOVS Rsrc, Rdst	0	0	0	0	1	0	0	00	0	0	0	0	0	0	0	0	0	0	0	0	0	11	00	1	0


Reset
SWINT
HWINT
RET/RTI
MEMOP & PCWrite
ConditionalJMP
ID/EX MemRead
ID/EX JMP/CALL/INT
Hazard Detection Unit
ID/EX Rdst
Rsrc1
Rsrc2
RET & RTI & HLT
Control Unit
JMPType[1:0]
IF/ID
ID/EX
EX/MEM
MEM/WB
WB
M
EX
Flush
+1
+2
INTIndex
LoadToPC FSM
EN
LoadPC
PCWrite
INTINDEX
JMP/CALL
RET/RTI/ LoadPC
PC
MemWrite
Memory
Write Data
Read Data
Address
MemRead
TOCCRREG
SWP
Imm32
Read Register1
Read Register2
Write Register
Write Data
Read data 1
Read data 2
Registers
RegWrite
Inst[23:21]
Inst[20:18]
Rsrc & Rdst
Rsrc2
Rsrc1
Funct
Imm16
Sign Extend
Immediate/Offset
Rdst
MemToReg
MemOp
StackOpType[1:0]
ALURes/In-port
PCStore
StackOp
STOREDATA
SP
PC
CCR
MEM/WBRegWrite
EX/MEMRegWrite
EX/MEMRdst
Forwarding Unit
Rt
Rs
ALU
ALUSrc
OutOp
ALU Control
Out Port
In Port
InputOp
EX/MEM RTI
(Used for moving original CCR and RTI CCR)
Disable ALU to ALU
Memory
ALU
EX/MEM RTI
CCR
Stack Pointer
Black Box
Idle
INT
interruptIndex
EX/MEM MemOp
LoadPC
interruptIndex
iF/ID EN
MemOp
IDLE: LoadPC = 0