Cairo University - Faculty of Engineering
CCE - Fall 2024

Computer Architecture Project
Phase 2

Students:

1. Amira Mohamed Essam Ahmed Abdelghany 1210200
2. Farida Ahmed Mohamed Moukhtar 1210276
3. Mariam Mahrous Mohamed Abd El Aal 1210301
4. Menna Salah Mohamed AlAwadly 1210032

Instruction format of your design

OpCode <15:11>	Src1 <10:8>	Src2 <7:5>	Dist <4:2>	Imm/offset? <1>	index <0>

Opcode for each instruction:

Instruction	OpCode (Type) <15:14>	OpCode (SubType) <13:11>
One Operand		
NOP	00	000
HLT	00	001
SETC	00	010
NOT	00	011
INC	00	100
OUT	00	101
IN	00	110
Two Operands		
MOV	01	000
ADD	01	001
SUB	01	010
AND	01	011
IADD	01	100

Memory Operations		
PUSH	10	000
POP	10	001
LDM	10	010
LDD	10	011
STD	10	100
Branch and Change of Control Operations		
JZ	11	000
JN	11	001
JC	11	010
JMP	11	011
CALL	11	100
RET	11	101
INT	11	110
RTI	11	111

Control Signal	Description
Mem_Read	Whether instruction will read from memory or not
Mem_Write	Whether instruction will write to memory or not
Mem_Write_Sel	Controls Source of Data written to memory (RSRC1 0 or PC Adder 1)
WEN_out	Write Enable Out: controls write enable of output port
Reg_Write	Controls Write Enable of register file
Reg_Write_Sel	Controls Source of Data written to register file (ALU 00 or Imm 01 or Memory 10 or Input Port 11)
ALU_Src	Controls second input to ALU (Register 0 or Immediate Value 1)
ALU_Sel	ALU selector: Controls ALU Operation (ADD 000, SUB 001, NOT 010, AND 011, INC 100, MOV 101, SETC110, RTI 111 restore flags)
JMP = pc_chng 01	Indicate unconditional branch (to take right PC)
JMP_Cond	Indicate conditional branch (Nothing 00, zero 01, Negative 10, Carry 11)
CCR_Write	Controls Write Enable of CCR : Carry, Negative, Zero (LSB)
SP_plus	In case of pop increment SP
SP_minus	In case of push decrement SP
Freeze	From CU to PC, when it is one, HLT
Ex_Res	Flag for Exceptions and Interrupts that controls that next instruction comes directly accessed from IM Exception: Empty Stack 10, Invalid Address 11 INT reserved flags: read 0+index
PC_chng	Controls how PC will change (PC_new 00, Register 01, Data Memory 10, Instruction Memory 11)

Inst	Mem_ Read	Mem_ Write	Mem_ Write_ Sel	WEN _out	Reg_ Write	Reg_ Write _Sel	ALU_ Src	Alu_ Sel	JMP	JMP _Co nd	CCR_ Write	SP_ plus	SP_ minus	Freeze	Ex_Res	PC_ Chng
NOP	0	0	x	0	0	xx	x	xxx	0	00	000	0	0	0	xx	00
HLT	0	0	x	0	0	xx	x	xxx	0	00	000	0	0	1	xx	00
SETC	0	0	x	0	0	xx	x	110	0	00	100	0	0	0	xx	00
NOT	0	0	x	0	1	00	x	010	0	00	011	0	0	0	xx	00
INC	0	0	x	0	1	00	x	100	0	00	111	0	0	0	xx	00
OUT	0	0	x	1	0	xx	x	xxx	0	00	000	0	0	0	xx	00
IN	0	0	x	0	1	11	x	xxx	0	00	000	0	0	0	xx	00
MOV	0	0	x	0	1	00	x	101	0	00	000	0	0	0	xx	00
ADD	0	0	x	0	1	00	0	000	0	00	111	0	0	0	xx	00
SUB	0	0	x	0	1	00	0	001	0	00	111	0	0	0	xx	00
AND	0	0	x	0	1	00	0	011	0	00	011	0	0	0	xx	00
IADD	0	0	x	0	1	00	1	000	0	00	111	0	0	0	xx	00
PUSH	0	1	0	0	0	xx	x	xxx	0	00	000	0	1	0	xx	00
POP	1	0	x	0	1	10	x	xxx	0	00	000	1	0	0	10 ?	00
LDM	0	0	x	0	1	01	x	xxx	0	00	000	0	0	0	xx	00
LDD	1	0	x	0	1	10	1	000	0	00	000	0	0	0	11 ?	00
STD	0	1	0	0	0	xx	1	000	0	00	000	0	0	0	11 ?	00

Inst	Mem_ Read	Mem_ Write	Mem_ Write_ Sel	WEN _out	Reg_ Write	Reg_ Write _Sel	ALU_ Src	Alu_ Sel	JMP	JMP _Co nd	CCR_ Write	SP_ plus	SP_ minus	Freeze	Ex_Res	PC_ Chng
JZ	0	0	x	0	0	xx	x	xxx	0	01	001	0	0	0	xx	00/01
JN	0	0	x	0	0	xx	x	xxx	0	10	010	0	0	0	xx	00/01
JC	0	0	x	0	0	xx	x	xxx	0	11	100	0	0	0	xx	00/01
JMP	0	0	x	0	0	xx	x	xxx	1	00	000	0	0	0	xx	01
CALL	0	1	1	0	0	xx	x	xxx	0	00	000	0	1	0	xx	01
RET	1	0	x	0	0	xx	x	xxx	0	00	000	1	0	0	xx	10
INT	0	1	1	0	0	xx	x	xxx	0	00	000	0	1	0	00/01	11
RTI	1	0	x	0	0	xx	x	111	0	00	111	1	0	0	xx	10

- Ay haga sp hasten wara b3d haram olna
- Ret ht reflect b3d el wb
- Last bit hya el index and set ex_res at memory according to control signals and pc_chng
- Sp plus or sp minus = zero and mem_write = 1 -> STD

General Notes:

- Unless otherwise stated, the signal comes from the control unit.
- **Static Prediction**: The static prediction is set to "Not Taken."
- **PC Change Handling**: If the Pc_Chng does not select the new PC, a flush will occur.
- **Freeze Logic**: The freeze condition will be handled with the logic: if rising_Edge(clk) and not(freeze).
- **Immediate Value Handling**: The immediate value fetched during the Fetch/Decode stage may or may not be needed depending on the instruction. If the immediate value is not required, we do not need to fetch the second instruction. Alternatively, both instructions can always be sent for processing after the fetch/decode stage.

Pipeline Stages Design:

Pipeline registers details (Size, Input, Connection, …)

Register	Size	Stored Values	Input Components	Output Components
Fetch/Decode	64 bits	PC Input Port Data Immediate Value Current Instruction	Reset (Signal) Flush Unit Instruction Memory Input Port	Control Unit Decode/Execute Sign Extend Register File
Decode/Execute	83 bits 22 bits ctrl	PC RDST Reg1 Data Reg2 Data Immediate Input Port Data	Fetch/Decode (PC, RDST, Input Port Data) Register File (Reg1 Data, Reg2 Data) Sign Extend Flush Unit	Execute/Memory 2 Mux to ALU
Execute/Memory	86 bits 22 bits ctrl	PC RDST Reg1 Data Input Port Data ALU Result Immediate CCR	Decode/Execute ALU	Memory/WriteBack Mux to Data Memory
Memory/WriteBack	83 bits 22 bits ctrl	Read Data ALU Result RDST Output Port Data Immediate Input Port Data	Execute/Memory Data Memory	Output Port Mux to Register File

- This data is subject to change while implementation.

Pipeline hazards and solution:

Type	Source	Detection	Solution
Data Hazard	RAW	Hazard Detection Unit	- Forward data using Forwarding Unit
Data Hazard	Load Use (Can be caused by LDD and POP instructions)	Hazard Detection Unit	- Stall (using NOP) until data is prepared from memory - Forward data using Forwarding Unit
Control Hazard	Branching	From Op_Code and Flags in case of conditional branch	- Prediction is always set to Not Taken - A Flush Unit is added

Other Signal	Description
Flush	Detect which pipeline registers to flush (0X No flush, 10 Fetch/Decode, 11 Fetch/Decode and Decode/Execute) If Flush[0]=1 -> flush control unit
Hazard_detection_Signal	Whether there is hazard or not 00 NO Hazard / 10 ALU Hazard / 11 Memory Hazard
FWD_Signal	Mainly Hazard_detection_Signal if there is hazard , if no hazard OR its second bit to the alu_src (00 Src1 or Src2 / 01 IMM (Case the second mux only) / 10 ALU Result / 11 Memory Result)

Notes:
Flush: no flush at execute/memory because no exceptions detected there
Sp_plus and sp_minus performed after decode
JMP, CALL, RET, INT, RTI, Stack Exception : FLUSH Fetch/Decode
JMP_Cond, Load Exception : FLUSH Fetch/Decode and Decode/Execute
Reserved Flag should be removed because we will not be able to do recursive interrupts, instead, we should push them in the last 4 bits of stack because

first 12 bits are for the PC address (which is 12 bits because 4k is 2^12)

-- bit 23: Mem_Read
-- bit 22: Mem_Write
-- bit 21: Mem_Write_Sel
-- bit 20: WEN_out
-- bit 19: Reg_Write
-- bit 18: Reg_Write_Sel (1)
-- bit 17: Reg_Write_Sel (0)
-- bit 16: ALU_Src
-- bit 15: ALU_Sel (2)
-- bit 14: ALU_Sel (1)
-- bit 13: ALU_Sel (0)
-- bit 12: JMP
-- bit 11: JMP_Cond (1)
-- bit 10: JMP_Cond (0)
-- bit 9: CCR_Write (2)
- - bit 8: CCR_Write (1)
-- bit 7: CCR_Write (0)
-- bit 6: SP_plus
-- bit 5: SP_minus
-- bit 4: Freeze
-- bit 3: Ex_Res (1)
-- bit 2: Ex_Res (0)
-- bit 1: PC_Chng (1)
-- bit 0: PC_Chng (0)

NOTE: IN CASE OF PUSH AND POP ALU SEL = 0

Comparator in execute w ha5od mno el Ex_Res w el pc_chng_new
Freeze mn el decode

****We removed reserved flag CCR