Cairo University
Faculty of Engineering
Computer Engineering Department

5-Stage Pipeline Processor Architecture

Team # 17 Members:

Radwa Samy Khattab

Abdelrahman Mohamed Badr

Mohamed Ramzy

Mai Mostafa

I. Instruction Register (IR) & Op-codes

1. General Registers

PC & SP are 32-bit registers. Initial value of SP = 2^31 -1.

Reg	code
R0	000
R1	001
R2	010
R3	011
R4	100
R5	101
R6	110
R7	111

2. One-Operand Operations

N.B. In case of NOP, to just increment the PC, Rdst bits are ignored (not interpreted to anything)

Operation	opcode
NOP	000
NOT	001
INC	010
DEC	011
OUT	100
IN	101

3. Two-Operand Operations

Oper.	op	IR					
SWAP	000	001	000	3-bits Rdst	XXX	3-bits R src1	X
ADD	001	001	001	3-bits Rdst	3-bits Rsrc2	3-bits Rsrc1	X
IADD	010	001	010	3-bits Rdst	XXX	3-bits Rsrc1	X
		16-bits immediate value					
SUB	011	001	011	3-bits Rdst	3-bits Rsrc2	3-bits Rsrc1	X
AND	100	001	100	3-bits Rdst	3-bits Rsrc2	3-bits Rsrc1	X
OR	101	001	101	3-bits Rdst	3-bits Rsrc2	3-bits Rsrc1	X
SHL	110	001	110	3-bits Rdst	XXXX XXX		
		16-bits immediate value					
SHR	111	001	111	3-bits Rdst	XXXX XXX		
		16-bits immediate value					

4. Memory Operations

Oper.	op	IR				
PUSH	000	010	000	3-bits Rdst	XXXX XXX	
POP	001	010	001	3-bits Rdst	XXXX XXX	
LDM	010	010	010	3-bits Rdst	XXXX XXX	
		16-bits immediate value				
LDD	011	010	011	3-bits Rdst	XXX	<M.S. 4-bits of EA> EA [19:16]
		<Rest of Effective Address> i.e. EA [15:0]				
STD	100	010	100	3-bits Rdst	XXX	<M.S. 4-bits of EA> EA [19:16]
		<Rest of Effective Address> i.e. EA [15:0]				

5. Branch and Change of Control Operations

011	3-bits opcode	3-bits Rdst	XXXX XXX

N.B. In case of RET & RTI, Rdst bits are ignored (not interpreted to anything)

Operation	opcode
JZ	000
JMP	001
CALL	010
RET	011
RTI	100

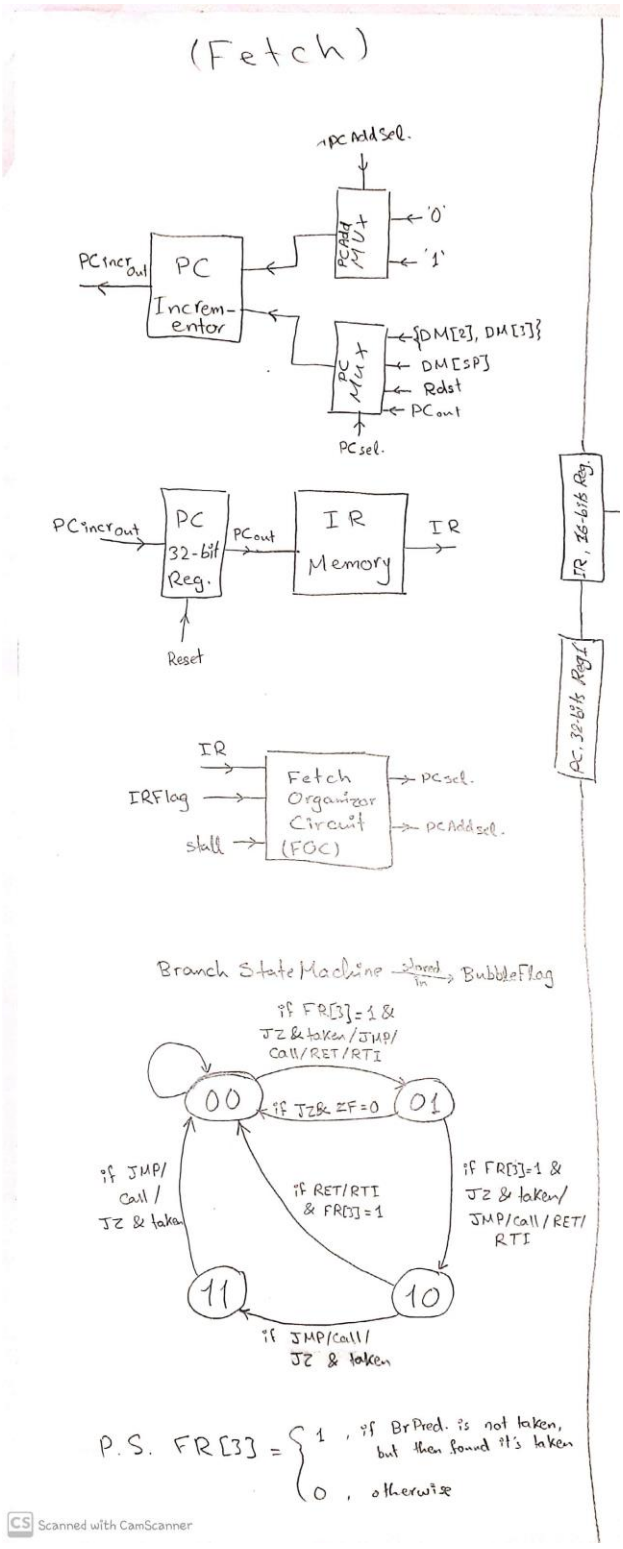
6. Input Signals

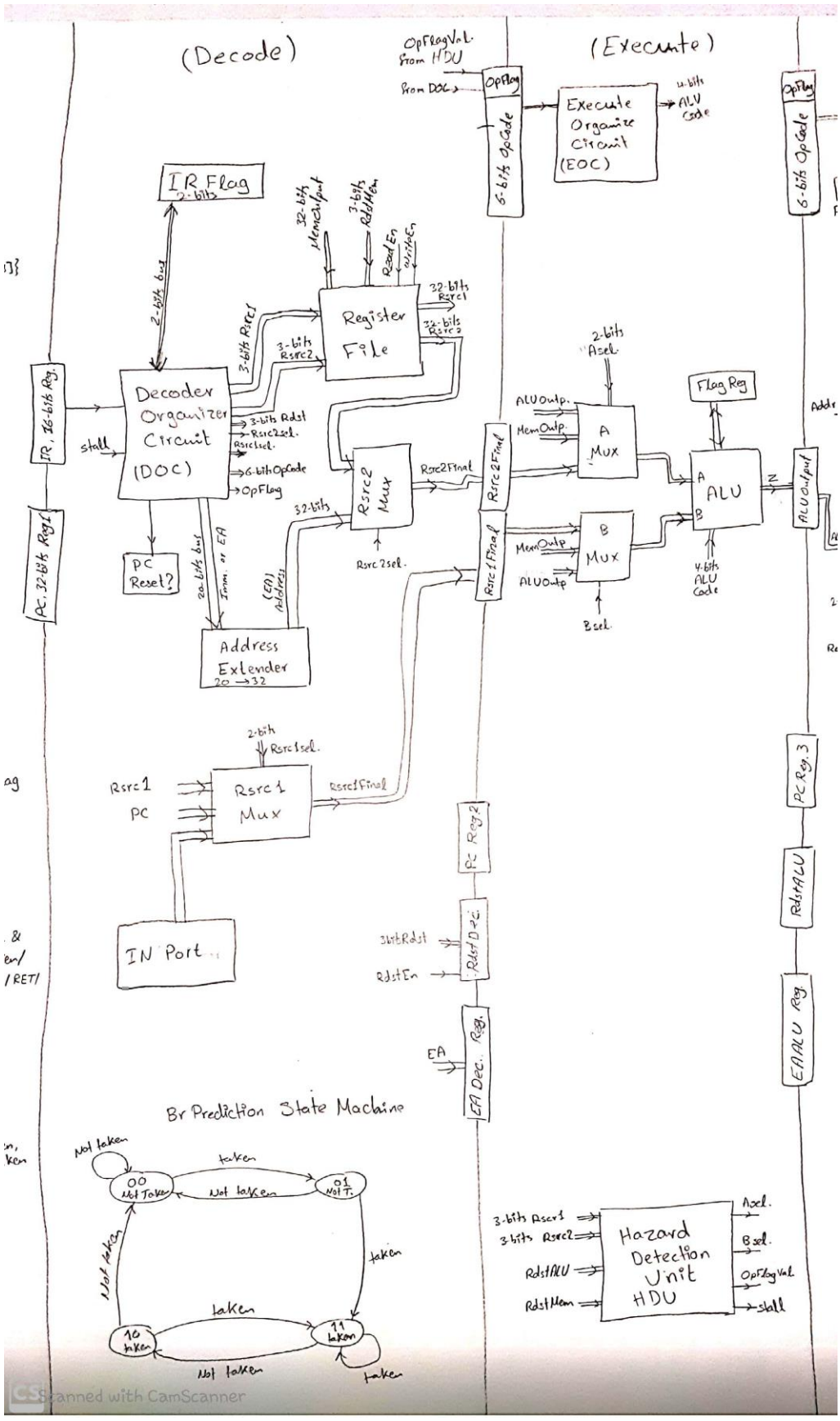
100	1-bit opcode	10	XXXX XXXX XX

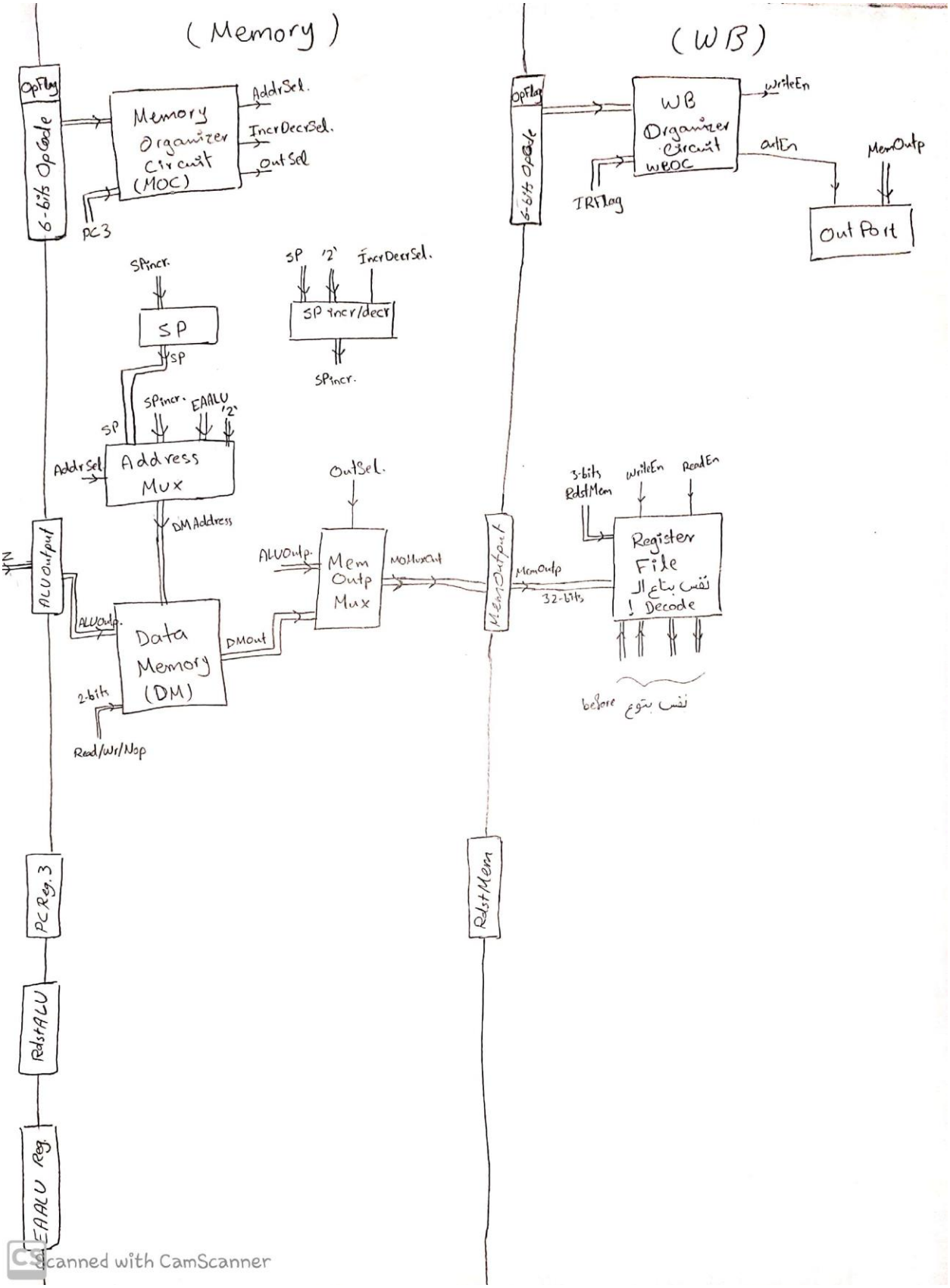
Operation	opcode
Reset	0
Interrupt	1

II. Schematic Diagram & Data Flow

Note in design: Enables are not all defined, but there's surely an enable to each component, and controlled by the O.C. of that stage.






(Memory)
(WB)
OpFlag
6-bits OpCode
Memory Organizer Circuit (MOC)
AddrSel.
IncrDecrSel.
OutSel
PC3
WB Organizer Circuit WBOC
writeEn
outEn
MemOutp
IRFlag
Out Port
SPincr.
SP
SP incr/decr
'2'
SP
Address Mux
EAALU
AddrSel
DM Address
ALUOutput
ALUOutp.
Data Memory (DM)
2-bits
Read/Wr/Nop
OutSel.
Mem Outp Mux
MOMuxOut
MemOutput
3-bits RdstMem
writeEn
ReadEn
Register File نفس بتاع ال Decode
32-bits
before نفس بتوع
PC Reg. 3
RdstALU
EAALU Reg.
RdstMem
Scanned with CamScanner

- ALU Operations with their equivalent codes:
    1. NOP (no operation)
    2. NopA
    3. NopB
    4. NotA
    5. IncA
    6. DecA
    7. AswapB
    8. AaddB
    9. AsubB (A – B)
    10. AandB
    11. AorB
    12. AshlB (SHL A by B)
    13. AshrB (SHR A by B)

- Main Registers used: (without pipelining registers)
    1. PC register – 32-bits
    2. BubbleFlag register – 2-bits (for bubbles state machine to fix control hazards)
    3. IRFlag register – 2-bits (for indicating what the IR represents)
        - 00 → normal IR
        - 01 → Immediate Value
        - 10 → 16-bits of the Effective Address
    4. BrPrediction Register – 2-bits
        - As drawn in design, used in branch prediction.
    5. Temp Effective Address Register (stores 4-bits of EA, waits for rest of 16-bits, then pass it to Address Extender to turn it into 32-bits and send it further – found inside Address Extender block)
    6. Flag Register – 4-bits
        - FR[0] = Zero Flag
        - FR[1] = Negative Flag
        - FR[2] = Carry Flag
        - FR[3] = wrong prediction to be ‘not taken’ when it’s ‘taken’
    7. Stack Pointer (SP) – 32-bits, initialized with 2^31 – 1

- Memories
    1. IR Memory
        - 2 GB in size, PC starts from 0 and has up to 2^31 – 1 locations.
    2. Data Memory
        - 2 GB in size, separated to memory from location 0, and stack starting at end at location 2^31 – 1. (say 1GB preserved for each)
    3. Register File
        - Has 8 general-purpose registers
        - Takes 3-bits for Rsrc1, Rsrc2, Rdst (named RdstMem in our design)
        - Takes 32-bits of Rdst data (named MemOutput in our design)
        - Has 2-enables; one for read and one for write
        - Outputs 32-bits Rsrc1 & Rsrc2

- Control Units/Blocks:
    1. Fetch Organizers Circuit (FOC)
        - Acts as the control unit in Fetch Stage.
        - Takes IR, IR Flag & stall-bit as input.
        - Outputs two signals, PCsel., PCAddsel, PCenable, IRMemEn & MUX'es.
        - Handles Branch State Machine (BubbleFlag).
        - Explanation:
            - According to the IR, I can decide whether I need to increment the PC by 1 (default), or stall (add '0' to it), or if I'm branching, then according to the state machine too, I'd either stall or take data from Rdst, or last data in stack, or Data at location [2] & [3] in case of interrupt.
            - These options are chosen from with two MUX'es.
            - Some notes regarding this block:
                - PC register outs on level 1 of the clock, and so is the IR & PCReg1
                - IRReg is written in on level 0, and so is PCincrOut.
                - In Branch S.M., I should reset FR[3] when get back to 00

    2. Decoder Organizer Circuit (DOC)
        - Acts as the control unit in Decode Stage.
        - Takes stall-bit, IR, & IRFlag as input.
        - Outs enables for MUX'es, Address Extender & Registers. Outs 3-bits of Rsrc1, Rsrc2, & Rdst. Sets PCReset-bit (which is connected to reset of PC register in

Fetch). Outs Immediate Value or EA to Address Extender Module (which includes a temp register for the data). Outs OpFlag (explained in pipeline), and 6-bits OpCode (IR[15:10]). And selectors for Rsrc1 & Rsrc2.

- Handles IR Flag.
- Handles Branch Prediction State Machine. (that's only when in JZ instruction)

3. Execute Organizer Circuit (EOC)
    - Takes 6-bits OpCode & generates 4-bits ALUCode accordingly.
    - Takes OpFlag, which is used as an enable to ALU
        - If OpFlag = 0, ALU works as it should
        - If OpFlag = 1, ALU does nothing (enable = 0)
    - Outs ASel. & BSel. For both MUX'es and their enables

4. Memory Organizer Circuit (MOC)
    - Takes 6-bits OpCode, & OpFlag
        - If OpFlag = 1, all enables are off—bubble
        - If OpFlag = 0, everything works normally
        - If I knew from OpCode I don't need Mem, it just outs ALUOutp. through MemOutp.
    - Takes PC register (actually from pipelining stages .. explained later)
    - Outs AddrSelector, IncrDecrSel, OutSel ➔ all are selectors to three MUX'es.
    - Out also enables to all registers & MUX'es in this stage.
    - Outs Enable to Data Memory (DM)
    - SP Incr./Decr. Block
        - Decides to increment or decrement SP by '2' sent to it, and outs SPincr.
    - AddressMux
        - Selects which address to enter the DM
            - SP, SPincremented, EA, or '2' (in case of interrupt)

5. Write Back Organizer Circuit (WBOC)
    - Takes OpFlag & 6-bits OpCode (to decide if WB is needed or not)
        - If OpFlag = 1 – bubble
    - Takes IRFlag
    - Outs WriteEnable to the Register File, and OutEnable that outputs on Out Port

III. Pipeline Stage Design

- Pipelining Registers (in-between stages)
    1. In Fetch/Decode interconnection
        - PC1 Register – 32-bits
        - IR Register – 16-bits

    2. In Decode/Execute interconnection
        - OpFlag – 1-bit
            - Basically this is set by DOC when I want to insert a bubble .. all stages ahead don't do anything if OpFlag = 1
        - OpCode – 6-bits
            - IR[15:10] that can indicate instructions uniquely.
            - Used in ahead stages to define the instruction at hand.
        - Rsrc1Final – 32-bits
            - Final value that Execute will operate on as A-operand.
            - At some cases, A can be a value that just needs to be passed on to further stages than execute.
        - Rsrc2Final – 32-bits
            - Final value that Execute will operate on as B-operand.
            - At some cases, B can be a value that just needs to be passed on to further stages than execute.
        - PC2 Register – 32-bits
            - Value that was stored in PC1 Reg. in previous interconnection.
        - RdstDec Register – 32-bits
            - 3-bits of Rdst, which are just IR[9:7]
        - EADec Register – 32-bits
            - Effective Address after being outed from Address Extender

    3. In Execute/Memory interconnection
        - OpFlag – 1-bit
            - Taken from OpFlag in previous interconnection
        - OpCode – 6-bits
            - Taken from OpCode in previous interconnection
        - ALUOutput – 32-bits
            - Output from ALU Operation

- PC3 Register – 32-bits
  - Value that was stored in PC2 Reg. in previous interconnection.
- RdstALU Register – 32-bits
  - Value that was stored in RdstDec Reg. in previous interconnection.
- EAALU Register – 32-bits
  - Value that was stored in EADec Reg. in previous interconnection.

4. In Mem/WB Interconnection
   - OpFlag – 1-bit
     - Taken from OpFlag in previous interconnection
   - OpCode – 6-bits
     - Taken from OpCode in previous interconnection
   - MemOutput – 32-bits
     - Output from Memory Stage, which is what's to be stored in Register File, if any at all
   - RdstMem Register – 3-bits
     - 3-bits indicating Rdst to write back in.

- Hazards
  1. Data Hazards – handled by Hazard Detection Unit
     - We Made Full-Forwarding, such that inputs to ALU are not directly from Decode stage, but there's a MUX at each A & B that chooses from either the input from Decode Stage, Last value stored in ALU-output register or last value stored in MemOutput register.
     - This way performs ALU-ALU forwarding & Memory-ALU forwarding
     - Example:

Memory-ALU	add R0, r1, r2	F	D	E	M	W					
ALU-ALU	sub r3, r1, r5		F	D	E	M	W				
	add R5, r0, r3			F	D	E	M	W			
	inc r2	F	D	E	M	W					
ALU-ALU	push r2		F	D	E	M	W				
	pop r5			F	D	E	M	W			
bubble/Memory-ALU	inc r5				F	D	D	E	M	W	
ALU-ALU	dec r5						F	D	E	M	W
	pop r6	F	D	E	M	W					
	inc r5		F	D	E	M	W				
Memory-ALU	dec r6			F	D	E	M	W			
	pop r5	F	D	E	M	W					
bubble/Memory-ALU	inc r5		F	D	D	E	M	W			
ALU-ALU	add r2, r4, r5				F	D	E	M	W		

2. Load-Data Hazards – handled by Hazard Detection Unit
    - Hazard Detection Unit takes 3-bits of Rsrc1 & Rsrc2 from Decode Stage, and 3-bits in RdstALU register & 3-bits in RdstMem register. Through those it can define the hazard that may happen, and accordingly takes action, through bubble (sets stall & forces 1 on OpFlag) or ALU-ALU forwarding or Mem-ALU forwarding.
    - Stall output is the one that handles Load-Data Hazards by stalling the pipeline and inserting bubbles. This stall flag is connected to the DOC & FOC, and in case it's 1, the DOC sets the OpFlag to 1, telling other stages to stall, and FOC stalls fetching too.
    - Examples:

insert bubble	LDD r3, 412	F	D	D	E	M	W	
	add r5, r3, r4			F	D	E	M	W

        - Generally, with LDD, LDM & STD, if same register used right after the current instruction, a bubble is inserted. If the same register is used after 1 instruction, then it can be handled by Memory-ALU forwarding.

3. Control Hazards
    - These are handled in Fetch through FOC and the Branch State Machine as stated before.

4. Structural Hazards
    - Won't occur because we're always writing in registers at level 0 and read in level 0, except in case of Register File, it's vice-versa.
    - Either way, memories won't be used by two stages at same time.