Cairo University
Faculty of
Engineering

Department of
Computer and
Communications
Engineering
CMPN 301 – Spring 2021

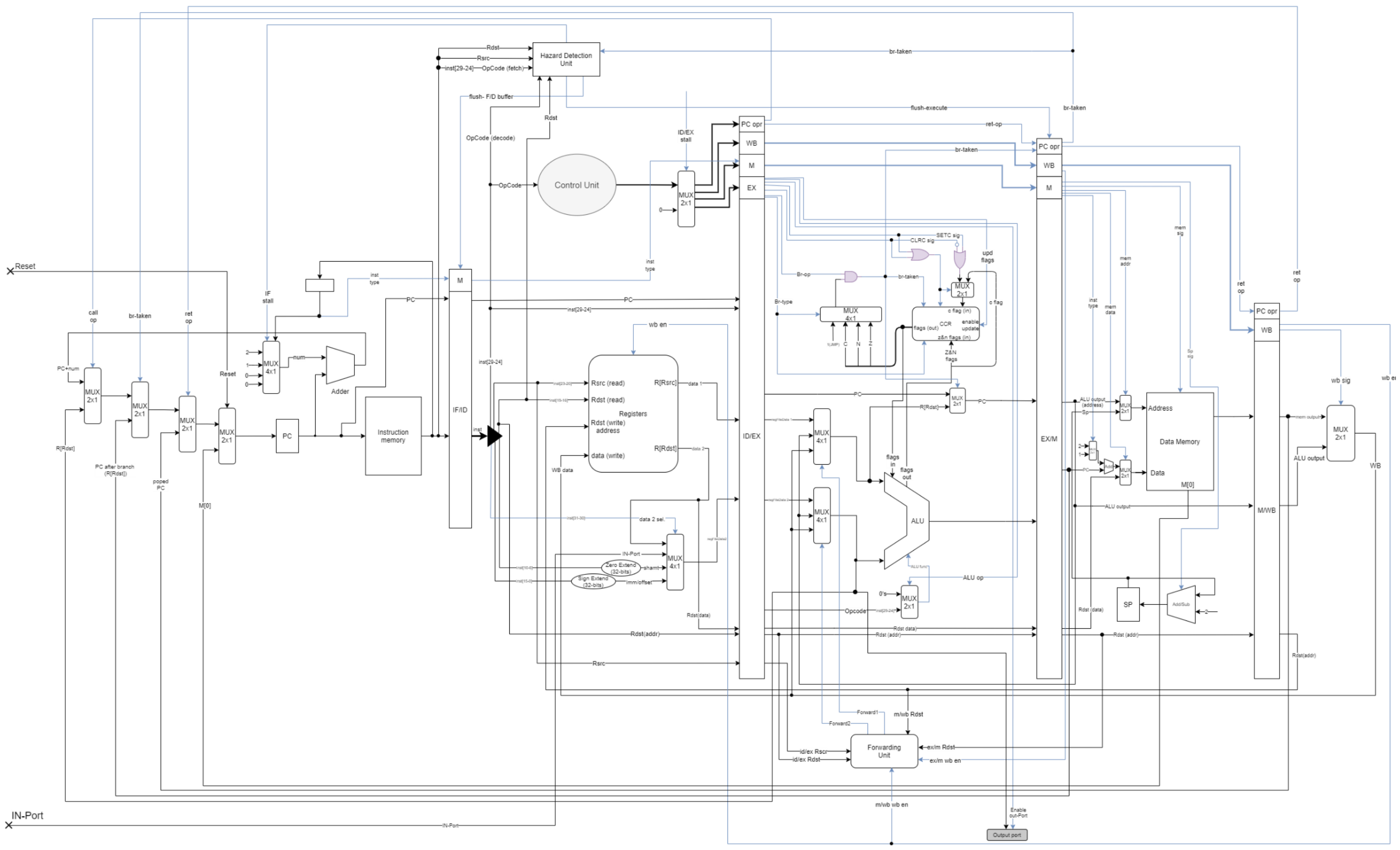
CMPN [301] Project Phase 2

Prepared by (team 4):

- Adham Mahmoud 1170115
- Kareem Mohamed 4180299
- Osama Yahia 1170141
- Omar M. Magdy 1163408

Submitted at: 16-May-2021

Schematic Design



Instruction format

R-Type Instructions (16 bit)

Instruction	Opcode [31-24]	R source [23-20]	R dist [19-16]
NOT Rdst	00010001	Rdst	Rdst
INC Rdst	00010010	Rdst	Rdst
DEC Rdst	00010011	Rdst	Rdst
OUT Rdst	00010100	Rdst	Rdst(X)
IN Rdst	01010101	Rdst	Rdst
MOV Rsrc, Rdst	00011110	Rsrc	Rdst
ADD Rsrc, Rdst	00011111	Rsrc	Rdst
SUB Rsrc, Rdst	00011000	Rsrc	Rdst
AND Rsrc, Rdst	00011001	Rsrc	Rdst
OR Rsrc, Rdst	00011010	Rsrc	Rdst

R-Type Instructions (32 bit)

Instruction	Opcode [31-24]	R source [23-20]	R dist [19-16]	Reserved [15-11]	Shamt [10-6]	Reserved [5-0]
SHL Rsrc, imm	10010110	Rsrc	1’s	0’s	Imm	0’s
SHR Rsrc, imm	10010111	Rsrc	1’s	0’s	Imm	0’s

I-Type Instruction (32 bit)

Instruction	Opcode [31-24]	R source [23-20]	R dist [19-16]	Offset/Imm [15-0]
IADD Rdst, imm	11100111	Rdst	Rdst	Imm
LDM Rdst, imm	11100100	1's	Rdst	Imm
LDD Rdst, offset (Rsrc)	11101101	Rsrc	Rdst	Offset
STD Rdst, offset (Rsrc)	11101110	Rsrc	Rdst	Offset

I-Type Instruction (16 bit)

Instruction	Opcode [31-24]	R source [23-20]	R dist [19-16]
PUSH Rdst	00100010	1's	Rdst
POP Rdst	00100011	1's	Rdst

J-Type Instruction (16 bit)

Instruction	Opcode [31-24]	R source [23-20]	Reserved [19-16]
JZ Rdst	00111001	Rdst	1's
JN Rdst	00111010	Rdst	1's
JC Rdst	00111011	Rdst	1's
JMP Rdst	00111100	Rdst	1's
CALL Rdst	00110101	Rdst	Rdst
RET	00110110	1's	1's

Special Instruction (16 bit)

Instruction	Opcode [31-24]	Reserved [23-16]
NOP	00000000	1’s
SETC	00000001	1’s
CLRC	00000010	1’s

*Opcode [31-30]: acts as a selector in the decode stage, to select what is the second data to pass to the execution stage.

*Opcode [29-28]: determines the instruction type.

Control Unit Signals

	Opcode Inst [29-24]	Br-op	Br-type	CLRC sig	SETC sig	Enable out-Port	ALU op	upd flags	mem addr	mem data	mem sig	SP sig	ex/m wb en	wb sig	wb en	call op	ret op
NOT Rdst	*010001*	0	XX	0	0	0	1	1	X	X	00	00	1	1	1	0	0
INC Rdst	*010010*	0	XX	0	0	0	1	1	X	X	00	00	1	1	1	0	0
DEC Rdst	*010011*	0	XX	0	0	0	1	1	X	X	00	00	1	1	1	0	0
OUT Rdst	*010100*	0	XX	0	0	1	0	0	X	X	00	00	X	X	0	0	0
IN Rdst	*010101*	0	XX	0	0	0	1	0	X	X	00	00	1	1	1	0	0
MOV Rsrc, Rdst	*011110*	0	XX	0	0	0	1	0	X	X	00	00	1	1	1	0	0
ADD Rsrc, Rdst	*011111*	0	XX	0	0	0	1	1	X	X	00	00	1	1	1	0	0
SUB Rsrc, Rdst	*011000*	0	XX	0	0	0	1	1	X	X	00	00	1	1	1	0	0
AND Rsrc, Rdst	*011001*	0	XX	0	0	0	1	1	X	X	00	00	1	1	1	0	0
OR Rsrc, Rdst	*011010*	0	XX	0	0	0	1	1	X	X	00	00	1	1	1	0	0
SHL Rsrc, imm	*010110*	0	XX	0	0	0	1	1	X	X	00	00	1	1	1	0	0
SHR Rsrc, imm	*010111*	0	XX	0	0	0	1	1	X	X	00	00	1	1	1	0	0
IADD Rdst, imm	*100111*	0	XX	0	0	0	1	1	X	X	00	00	1	1	1	0	0
PUSH Rdst	*100010*	0	XX	0	0	0	0	0	1	1	01	01	X	X	0	0	0
POP Rdst	*100011*	0	XX	0	0	0	0	0	1	X	10	10	1	0	1	0	0
LDM Rdst, imm	*100100*	0	XX	0	0	0	1	0	X	X	00	00	1	1	1	0	0
LDD Rdst, offset (Rsrc)	*101101*	0	XX	0	0	0	1	0	0	X	10	00	1	0	1	0	0
STD Rdst, offset (Rsrc)	*101110*	0	XX	0	0	0	1	0	0	1	01	00	X	X	0	0	0
JZ Rdst	*111001*	1	00	0	0	0	0	0	X	X	00	00	X	X	0	0	0
JN Rdst	*111010*	1	01	0	0	0	0	0	X	X	00	00	X	X	0	0	0
JC Rdst	*111011*	1	10	0	0	0	0	0	X	X	00	00	X	X	0	0	0
JMP Rdst	*111100*	1	11	0	0	0	0	0	X	X	00	00	X	X	0	0	0
CALL Rdst	*110101*	0	XX	0	0	0	0	0	1	0	01	01	X	X	0	1	0
RET	*110110*	0	XX	0	0	0	0	0	1	X	10	10	X	X	0	0	1
NOP	*000000*	0	XX	0	0	0	0	0	X	X	00	00	X	X	0	0	0
SETC	*000001*	0	XX	0	1	0	0	0	X	X	00	00	X	X	0	0	0
CLRC	*000010*	0	XX	1	0	0	0	0	X	X	00	00	X	X	0	0	0

Control Signals details

br-type:
00 -> Z
01 -> N
10 -> C
11 -> 1
----
mem addr:
0 -> alu output
1 -> SP
----
mem data:
0 -> PC + 2
1 -> R[Rdst]
----
mem sig:
00 & 11 -> do nothing
01 -> store (take address & data)
10 -> load (take address)

SP sig:
00 & 11 -> do nothing
01 -> sub
10 -> add
----
wb sig
0 -> mem out
1 -> alu out
----
data 2 sel:
00 -> data 2
01 -> IN-Port
10 -> shamt
11 -> Imm/Offset

Br-taken (at ex stage):
0 -> PC
1 -> R[Rdst]

ALU op:
0 -> 0's
1-> inst[29-24]

Forwarding Unit

Inputs:

- id/ex Rsrc
- id/ex Rdst
- m/wb Rdst
- ex/m Rdst
- ex/m wb en
- m/wb wb en

Outputs:

- Forward1
- Forward2

Forward1 & Forward2 are the selectors for the 2 mux's that is connected to the ALU.

For forward1:

```
If( (id/ex Rsrc == ex/m Rdst) && (ex/m wb en) )          ALU – ALU forwarding
    Forward1 = 01
Else if( (id/ex Rsrc == m/wb Rdst) && (m/wb wb en) )     MEM – ALU forwarding
    Forward1 = 10
Else
    Forward1 = 00                 No forwarding
```

For forward2:

```
If( (id/ex Rdst == ex/m Rdst) && (ex/m wb en) )          ALU – ALU forwarding
    Forward2 = 01
Else if( (id/ex Rdst == m/wb Rdst) && (m/wb wb en) )     MEM – ALU forwarding
    Forward2 = 10
Else
    Forward2 = 00                 No forwarding
```

Hazard Detection Unit

Inputs:

- Memory Read.
- Branch Taken.
- Rs
- Rt

Outputs:

- ID/EX Stall
- IF Stall
- Flush Decode
- Flush Execute

- ID/EX Stall: This signal is used to pass zeros to the control signals in order to activate the NOP Instruction which is zeros.

- IF Stall: The stall signal is used to make the PC stalled (constant) specifically the fetch stage to stall the pipeline.

- Rt: Destination Register used for comparison to detect data dependency hazards.

- Rs: Source Register used for comparison to detect data dependency hazards.

- Flush decode: this signal is used to flush the decode buffer in case of call/jump/return instruction.

- Flush Execute: this signal is used to flush the decode buffer in case of call/jump/return instruction.

- Memory-read: the signal is used to know if memory is used for reading or not in the current instruction.

- Branch taken: this signal is used to know if a branch instruction was taken (condition is true), this is necessary for flushing buffers.

Scenarios

1- **Call** Instruction: This is a multi-cycle instruction, it takes 2 cycles to execute, one of them is "NOP", it's executed or finished in execute stage, we must flush the fetch and decode stages as we won't need their values.

2- **RET** Instruction: This is also a multi-cycle instruction that takes 4 cycles to execute, this is executed in the Write-Back stage, 3 cycles are dedicated to NOP, no flushing is needed here since we are going to stall the pipeline.

Pipeline Buffers

- Buffer between **Fetch** and **Decode** contains

Input to buffer	Output from buffer
- PC (32 bit) - Instruction bits (32 bit)	- PC (32 bit) - Instruction bits (32 bit)

IF/ID buffer is 64 bits.

- Buffer between **Decode** and **Execute** contains

Input to buffer	Output from buffer
- Pc operations signal which contain (Call op signal, return operation signal) (2 bits) - Write back signal (2 bits for Wb signal, Wb enable) - Execute signal (7 bits) - Memory signal (4 bits) - PC address (32 bits) - inst[29-24] (OP code)(6 bits)	- Pc operations signal which contain (Call op signal, return operation signal) (2 bits) - Write back signal (2 bits for Wb signal, Wb enable) - Execute signal (7 bits) - Memory signal (4 bits) - PC address (32 bits) - inst[29-24] (OP code)(6 bits)

- Data and Address of Rsrc (32 bits data, 3 bits address) - Data and Address of Rdst (32 bits data ,3 bits address) - The output of Mux which select which data will pass to Alu (Shift amount, Sign extend, In port data) (32 bit) - flush decode (1 bit)	- Data and Address of Rsrc (32 bits data, 3 bits address) - Data and Address of Rdst (32 bits data ,3 bits address) - The output of Mux which select which data will pass to Alu (Shift amount, Sign extend, In port data) (32 bit)

ID/EX buffer size is 156 bits.

- Buffer between **Execute** and **Memory** contains

Input to buffer	Output from buffer
- flush execute(1bit) - PC operation signal (3 bits) - Write back signal (2 bits) - Memory signal (4 bits) - PC Address (32 bits) - ALU output (32 bits) - Rdst data and address (35 bits)	- flush execute(1bit) - PC operation signal (3 bits) - Write back signal (2 bits) - Memory signal (4 bits) - PC Address (32 bits) - ALU output (32 bits) - Rdst data and address (35 bits)

EX/M buffer size is 109 bits.

- Buffer between **Memory** and **Write Back** contains

Input to buffer	Output from buffer
- PC operation signal (1 bit) - Writeback signal (2 bits) - Output data from memory (32 bits) - ALU Output (32 bits) - Rdst address (3 bits)	- PC operation signal (1 bit) - Writeback signal (2 bits) - Output data from memory (32 bits) - ALU Output (32 bits) - Rdst address (3 bits)

M/WB buffer size is 70 bits.