Computer Architecture

Report

Written by

Name	Section	BN
Mohamed Saad	2	14
Karim Taha	2	8
Essam Wisam Fouad	2	01
Ahmed Mahmoud	1	13

Under the supervision of
Dr. Mayada Hadhoud
Eng. Maheed Hatem

December 2021

CU Unit Design

One Operand Instructions

Instruction	OPSrc	TFAOI	ALUOP	AddSrc	MEMW	SPA	WF	PCSrc	WB	SaveF	ResF	PCC	JM	Write Out
NOP	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HLT	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SETC	0	0	SETC	0	0	0	0	0	0	0	0	0	0	0
NOT RDST	0	0	NOT	0	0	0	ALU	0	1	0	0	0	0	0
INC RDST	0	0	INC	0	0		ALU	0	1	0	0	0	0	0
OUT RDST	0	0	Forw.	0	0	0	0	0	0	0	0	0	0	1
IN RDST	0	1 (IN)	0	0	0	0	ALU	0	1	0	0	0	0	0

Two Operand Instructions

Instruction	OPSrc	TFAOI	ALUOP	AddSrc	MEMW	SPA	WF	PCSrc	WB	SaveF	ResF	PCC	JM	WriteOut
MOV	0	0	forward	0	0	0	ALU	0	1	0	0	0	0	0
ADD	0	0	add	0	0	0	ALU	0	1	0	0	0	0	0
SUB	0	0	sub	0	0	0	ALU	0	1	0	0	0	0	0
AND	0	0	and	0	0	0	ALU	0	1	0	0	0	0	0
IADD	1	0	add	0	0	0	ALU	0	1	0	0	0	0	0

Memory Instructions

Instruction	OPSrc	TFAOI	ALUOP	AddSrc	MEMW	SPA	WF	PC Src	WB	SaveF	ResF	PCC	JM	WriteOut
PUSH	0	0	0	SP	Write	Sub 1	0	0	0	0	0	0	0	0
POP	0	0	0	SP	0	Add 1	M	0	1	0	0	0	0	0
LDM	IMM	0	ForwardLower	SP	0	0	A	0	1	0	0	0	0	0
LDD	IMM	0	ADD	M	0	0	M	0	1	0	0	0	0	0
STD	IMM	0	ADD	M	Write	0	0	0	0	0	0	0	0	0

Branch Instructions

Instruction	OPSrc	TFAOI	ALUOP	AddSrc	MEMW	SPA	WF	PCSrc	WB	SaveF	ResF	PCC	JM	WriteOut
JZ Rdst	0	0	0	0	0	0	0	Reg	0	0	0	3	OP[2:0]	0
JN Rdst	0	0	0	0	0	0	0	Reg	0	0	0	3	OP[2:0]	0
JC Rdst	0	0	0	0	0	0	0	Reg	0	0	0	3	OP[2:0]	0
JM Rdst	0	0	0	0	0	0	0	Reg	0	0	0	3	OP[2:0]	0
Call Rdst	0	0	0	SP	1	sub2	0	Reg	0	0	0	3	OP[2:0]	0
Ret	0	0	0	Stack	0	add2	0	Data	0	0	0	3	OP[2:0]	0
Int index	0	0	0	SP	1	sub2	0	Data	0	1	0	3	OP[2:0]	0
RTI	0	0	0	Sp	0	add2	0	Data	0	0	1	3	OP[2:0]	0

Bits for each signal

OPSrc	TFAOI	ALUOP	AddSrc	MEMW	SPA	WF	PCSrc	WB	SaveF	ResF	PCC	JM	WriteOut	1st Bit
1	1	3	2	1	3	1	2	1	1	1	2	3	1	1

Instruction Memory Format

One Operand Instructions

Instruction	OpCode (__ ___)	Format
NOP	1 0	**OP-DST-SRC1-SRC2-IMM**
HLT	1 1	**OP-DST-SRC1-SRC2-IMM**
SETC	1 2	**OP-DST-SRC1-SRC2-IMM**
NOT RDST	1 3	**OP-DST-SRC1-SRC2-IMM**
INC RDST	1 4	**OP-DST-SRC1-SRC2-IMM**
OUT RDST	1 5	**OP-DST-SRC1-SRC2-IMM**
IN RDST	1 6	**OP-DST-SRC1-SRC2-IMM**

Two Operand Instructions

Instruction	OpCode (__ ___)	Format
MOV	2 0	**OP-DST-SRC1-SRC2-IMM**
ADD	2 1	**OP-DST-SRC1-SRC2-IMM**
SUB	2 2	**OP-DST-SRC1-SRC2-IMM**
AND	2 3	**OP-DST-SRC1-SRC2-IMM**
IADD	2 4	**OP-DST-SRC1-SRC2-IMM**

Memory Instructions

Instruction	OpCode (__ ___)	Format
PUSH	0 0	**OP-DST-SRC1-SRC2-IMM**
POP	0 1	**OP-DST-SRC1-SRC2-IMM**
LDM	0 2	**OP-DST-SRC1-SRC2-IMM**
LDD	0 3	**OP-DST-SRC1-SRC2-IMM**
STD	0 4	**OP-DST-SRC1-SRC2-IMM**

Branch Instructions

Instruction	OpCode (__ ___)	Format
JZ Rdst	3 0	**OP-DST-SRC1-SRC2-IMM**
JN Rdst	3 1	**OP-DST-SRC1-SRC2-IMM**
JC Rdst	3 2	**OP-DST-SRC1-SRC2-IMM**
JM Rdst	3 3	**OP-DST-SRC1-SRC2-IMM**
Call Rdst	3 4	**OP-DST-SRC1-SRC2-IMM**
Ret	3 5	**OP-DST-SRC1-SRC2-IMM**
Int index	3 6	**OP-IND-SRC1-SRC2-IMM**
RTI	3 7	**OP-DST-SRC1-SRC2-IMM**

Instruction Piece	No. of bits
OP	5
DST	3
SRC1	3
SRC2	3
IMM	16
Ind	2

Note that the first two bits of the instruction are reserved for special checks (before opcode)

Pipeline Hazards & Solutions

Control Hazards:

- In our architecture, we assume branches to be untaken. Depending on the instruction we might figure out that our prediction was or was not correct in either the execute or memory stages. If a misprediction is discovered in the execute stage we flush the two previous instructions and if it's discovered in the memory we stage we flush the three previous instructions. When both the instructions in memory and execute want to change PC, priority is given to that in memory (it's there first)

Data Hazards:

- We used Full-forwarding(ALU-ALU, Memory-ALU)
- Memory-Memory forwarding (to solve the load-store case)

Structural Hazards:

- For the structural hazard between decode and writeback, the latter is done in the first half cycle and the former is done in the second half cycle.
- We have also ran into another structural hazard regarding the stack to prevent reading/writing data. In the memory stage, we read the stack in the first half cycle and then if there's need to modify it then that's done in the second half cycle.

Buffer Details

Stage	Inputs and No. of bits
Fetch/Decode	Inst[15:0] + IM[15:0] + PC[31:0] = 64 bit
Decode/Execute	CS[23:0] + Imm[15:0] + Rsrc1[15:0] + Rsrc2[15:0] + &Rdst[2:0] + &Rsrc1[2:0] + &Rsrc2[2:0] + PC[31:0] + Ind[1:0] =115 bit
Execute/Memory	CS[23:0] + Rsrc1[15:0] + Result[15:0] + &Rdst[2:0] + &Rsrc1[2:0] + &Rsrc2[2:0] + PC[31:0] + Ind[1:0] = 99 bit
Memory/Write Back	Result[15:0] + data[15:0] + Rsrc1[15:0] + CS[23:0] + &Rds[2:0] + &Rsrc1[2:0] + &Rsrc2[2:0] = 81 bit

Thus, the buffer size is 128 bits.