5-stage Pipelined Processor Design Report

Team #4

Mohamed Shawky
SEC:2, BN:16

Remonda Talaat
SEC:1, BN:20

Evram Youssef
SEC:1, BN:9

Mahmoud Adas
SEC:2, BN:21

March 30, 2020

Contents

List of Figures

List of Tables

Part I

Introduction

0.1 System Overview

This document reports our design work of the 5-stage pipelined processor using Harvard architecture. We discuss the overall system blocks and connections, the functionalities of the different blocks and the hazard solutions.

0.2 Task Distribution

TODO: Task distribution table

Part II

Overall System

0.3 Overall System Design Schema

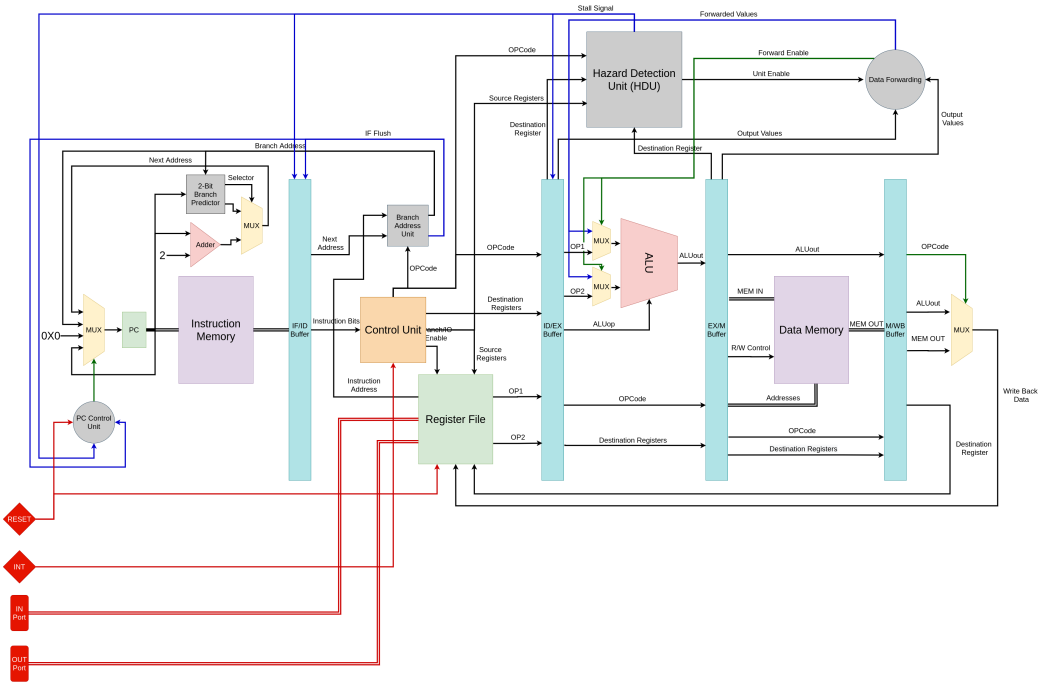

Figure 1: Overall System Design

TODO(1): Add branch predictor design
TODO(2): Add Interrupt handling design
TODO(3): Add stack pointer related instructions design
TODO(4): Add 32-bit fetch detection unit

0.4 Memory Specs

- We have 2 separate memory units, one for instructions and another for data and stack.
- Instruction Memory:
  - 2^{32} X 16 bits

  - 16-bit bus

- Data Memory:
  - 2^{32} X 16 bits
  - 32-bit bus
  - SP starts at 2^{32}-1

0.5 PC Control Unit

0.5.1 Inputs

- IF Flush (1 bit)
- Stall Signal (1 bit)
- RESET Signal (1 bit)

0.5.2 Outputs

- PC Mux Selectors (2 bits)

0.5.3 Logic

- If IF Flush == 1, Output = 01
- If RESET == 1, Output = 10
- If Stall == 1, Output = 11
- Else, Output = 00

0.6 Dynamic Branch Prediction

TODO: Add branch prediction details in both overall diagram and section

0.7 Branch Address Unit

0.7.1 Inputs

- PC Next Address (32 bits)
- Instruction Address (32 bits)
- OpCode (7 bits)

0.7.2 Outputs

- IF Flush (1 bit)
- Branch Address (32 bits)

0.7.3 Logic

- Check if OpCode is of a branch instruction, if true:
  - Check whether PC Next Address is equal to Instruction Address
  - If true:
    - IF Flush = 0, Branch Address = Instruction Address
  - If false:
    - IF Flush = 1, Branch Address = Instruction Address

0.8 Register File

0.8.1 Registers

- 8 general purpose registers
- Stack pointer (SP) register
- Program counter (PC) register

0.8.2 Inputs

- Dest Regs: $2X4$ bits (for destination selection)
- SRC Regs: $2X4$ bits (for source selection)
- WB values: $2X32$ bits (for write back values)
- RESET: 1 bit (for registers clear).
- Branch/IO: 2 bits (to determine whether the operation is IO or branch)
- IN Port: 32 bits (IO input port)

0.8.3 Outputs

- OP1: 32 bits (value of first operand)
- OP2: 32 bits (value of second operand)
- Instruction Address: 32 bits (value of branch address)
- OUT Port: 32 bits (IO output port)

0.8.4 Logic

The register selector acts like a decoder to select the required operation and the register on which the operation performed.

0.9 ALU

0.9.1 Inputs

- ALUop: 4 bits (refer to ALU Operations below)
- Operands: $2X32$ bits (2 input operands)

0.9.2 Outputs

- ALUout: 32 bits (operation result)

0.9.3 ALU Operations

- NOP – 0000
- INC – 0001
- DEC – 0010
- ADD – 0011
- SUB – 0100
- AND – 0101
- OR – 0110
- NOT – 0111
- SHL – 1000
- SHR – 1001

0.9.4 Logic

- ALU performs the operation and changes the CCR accordingly.
- The input operands of the ALU are multiplexed between forwarded data and register data, with selectors from data forwarding unit.

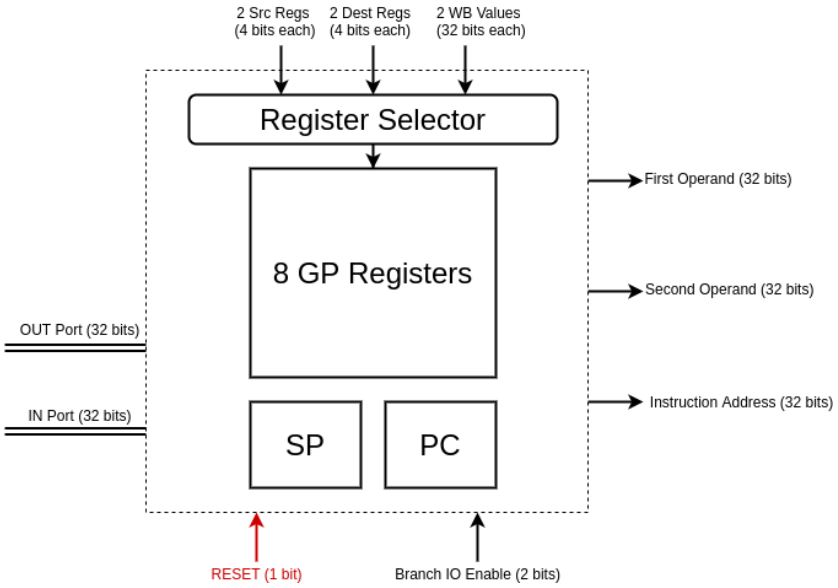

Figure 2: Register File Diagram

Part III

Instruction Format

0.10 One Operand Operations

- 4 bits (1111) for one operand instructions.
- 3 bits to define instruction.
- 3 bits for destination register.
- 1 bit to define the memory slots occupied by the instruction.
- Total of 11 bits, padded with 5 0's to fit 16 bits.

Table 1: One Operand Instruction Mapping

Operation	OpCode	Destination	16|32	Conditions
IN	1111000	000:111	0	————————
NOT	1111001	000:111	0	if !Rdst=0,Z=1 if !Rdst<0,N=1
INC	1111010	000:111	0	if Rdst+1=0,Z=1 if Rdst+1<0,N=1
DEC	1111011	000:111	0	if Rdst-1=0,Z=1 if Rdst-1<0,N=1
OUT	1111100	000:111	0	————————

0.11 Special Operations

- 16 0's to represent NOP (0000000000000000).

0.12 Two Operand Operations

- 4 bits to define instruction.
- 3 bits for each of Rsrc1, Rsrc2 and Rdst.
- 1 bit to define the memory slots occupied by the instruction.
- 16 bits for immediate values.
- Total of 14 bits in most cases with some exceptions mentioned below.

Table 2: Two Operand Instruction Mapping

Operation	OpCode	Rsrc1	Rsrc2	Rdst	imm	16|32	Conditions
SWAP	0001	000:111	—	000:111	—	0	———————— _
ADD	0010	000:111	000:111	000:111	—	0	if Result=0,Z=1 if Result<0,N=1
SUB	0011	000:111	000:111	000:111	—	0	if Result=0,Z=1 if Result<0,N=1
AND	0100	000:111	000:111	000:111	—	0	if Result=0,Z=1 if Result<0,N=1
OR	0101	000:111	000:111	000:111	—	0	if Result=0,Z=1 if Result<0,N=1
SHL	0110	000:111	—	—	16 bits	1	update carry flag
SHR	0111	000:111	—	—	16 bits	1	update carry flag
IADD	1000	000:111	—	000:111	16 bits	1	if Result=0,Z=1 if Result<0,N=1

0.13 Memory Operations

- 4 bits to define instruction.
- 3 bits for destination register.
- 1 bit to define the memory slots occupied by the instruction.
- 16 bits for immediate values.
- 20 bits for effective addresses.
- Total of 8 bits with no immediate values or effective addresses.
- Total of 24 bits with immediate values.
- Total of 28 bits with effective addresses.

Table 3: Memory Instruction Mapping

Operation	OpCode	Rdst	imm	EA	16|32	Conditions
PUSH	1001	000:111	—	—	0	————————
POP	1010	000:111	—	—	0	————————
LDM	1011	000:111	16 bits	—	1	————————
LDD	1100	000:111	—	20 bits	1	————————
STD	1101	000:111	—	20 bits	1	————————

0.14 Branch and Change Control Operations

- 4 bits (0000) for branching instructions.
- 3 bits to define instruction.
- 3 bits for destination register.
- 1 bit to define the memory slots occupied by the instruction.
- Total of 11 bits, padded with 5 0's to fit 16 bits.

Table 4: One Operand Instruction Mapping

Operation	OpCode	Destination	16|32	Conditions
JZ	0000001	000:111	0	————————
JMP	0000010	000:111	0	————————
CALL	0000011	000:111	0	————————
RET	0000100	—	0	————————
RTI	0000101	—	0	————————

Part IV

Control Signals

TODO: Add related control signals for each instruction

Part V

Pipeline Stages

0.15 Overview

TODO: Add an overview of how pipelining works in our system

0.16 IF/ID Buffer

TODO: Add buffer components

0.17 ID/EX Buffer

TODO: Add buffer components

0.18 EX/M Buffer

TODO: Add buffer components

0.19 M/WB Buffer

TODO: Add buffer components

Part VI

Pipeline Hazards and solutions

0.20 Hazards Detection and Handling

TODO: refactor this section and add more details

0.20.1 Structural Hazards

2 memory units, one for instructions and one for data. Both have the same specs *(mentioned below)*.
Also, to handle register file structural hazard, the write back will be in the first half of the clock cycle and the decode will be in the second half.

0.20.2 Data Hazards

Refer the *HDU* in 1

Stall

Occurs only at Decode stage, due to load use case.

- Fetch same instruction (don't increment the program counter).
- Latch IF/ID buffer with the same values.
- Freeze Decode stage.
- Clear ID/EX buffer.

Data Forwarding

- EX/MEM buffer $->$ Execute / Decode.
- ID/EX buffer $->$ Decode.

0.20.3 Control Hazards

- At Fetch stage, always check the branch predictor and calculate the next address accordingly.
- At Decode stage, check whether the OPCode is of a branch operation. If so, pass the address to the program counter and compare the correct address with the address of the counter to decide whether to flush the Fetch stage or not.

Flush

Occurs only at Fetch Stage, due to wrong branch prediction at Decode stage.

- Load new address in the program counter.
- Remove fetched instructions from IF/ID buffer.

Dynamic Branch Prediction

Hash table of some length containing Keys of branch addresses paired with 1-bit for branch prediction.