CSE 306 (Computer Architecture Sessional)
July 2023 Term

ALL LAB SECTIONS

---

1 PROBLEM DESCRIPTION

In this assignment, you will have to design, simulate (in Software), and implement (in Hardware) a **4-bit** processor that implements the MIPS instruction set. Each instruction will take 1 clock cycle to be executed. The length of the clock cycle will be long enough to execute the longest instruction in the MIPS instruction set. The main components of the processor are:

- Instruction Memory
- Data Memory
- Register File
- ALU
- Control Unit

Note that additional components such as Multiplexers, Adders etc. can be added as required by your design.

2 DESIGN SPECIFICATION

- The Address bus will have a size of 8-bits.
- The Data bus will have a size of 4-bits.
- A 4-bit *ALU* will be required, hence the name 4-bit MIPS.
- The *register file* must include the following temporary registers:

  $zero, $t0, $t1, $t2, $t3, $t4

  Each register will have a size of 4-bits. The assembly code that will be provided to simulate your design will use only the above-mentioned registers.
- The *control unit* should be micro-programmed. The control signals associated with the operations should be stored in a special memory (you can use a separate EEPROM for this purpose) unit as Control Words.
- All clocks required in the circuit must be provided from a single clock source. Each MIPS instruction should be fetched and executed in a single clock cycle.

- Your design will be evaluated based on accuracy (correct flow of execution for each implemented instruction), completeness (all assigned instructions have been implemented), and efficiency (minimizing the number of ICs used, automated assembler).

3 Instruction Set Description

Instruction ID	Category	Type	Instruction
A	Arithmetic	R	add
B	Arithmetic	I	addi
C	Arithmetic	R	sub
D	Arithmetic	I	subi
E	Logic	R	and
F	Logic	I	andi
G	Logic	R	or
H	Logic	I	ori
I	Logic	S	sll
J	Logic	S	srl
K	Logic	R	nor
L	Memory	I	sw
M	Memory	I	lw
N	Control-conditional	I	beq
O	Control-conditional	I	bneq
P	Control-unconditional	J	j

4 MIPS Instruction Format

* Our MIPS Instructions will be 16-bits long with the following three formats.

R-type	Opcode	Src Reg-1	Src Reg-2	Dst Reg
	4-bits	4-bits	4-bits	4-bits
S-type	Opcode	Src Reg-1	Dst Reg	Shamt
	4-bits	4-bits	4-bits	4-bits
I-type	Opcode	Src Reg-1	Src Reg-2/Dst Reg	Address/Immediate
	4-bits	4-bits	4-bits	4-bits
J-type	Opcode	Target Jump Address		0
	4-bits	8-bits		4-bits

5 Memory Considerations

For a single cycle implementation of MIPS instruction set, you need to have two separate memory units for instruction and data.

- Instruction Memory is accessed through the Program Counter (PC) register.
- Data Memory is accessed through address.

6 Instruction Set Assignment

The opcodes of the instruction will be between 0 to 15 based on the sequence of instruction id given below. Sequence ABCDEFGHIJKLMNOP means add instruction's opcode will be 0, addi instruction's opcode will be 1, sub instruction's opcode will be 2, and so on.

Group ID	Section A1	Section A2	Section B1	Section B2
1	JDNKLIMFEOPBGCHA	HBDMNEKCPLAIGJFO	GBFOLNAIEDJMKHCP	IOECDBPNKMGHJFLA
2	FLHGNIKAJBCMPEDO	KAGBLIPJNFMDCHEO	HOCIKJEBAFNLDGMP	HCAGNMIJDBOFEPKL
3	CLIJMHOEBKFGDNAP	KILHJDPNBMFGEACO	POJMBNDECAGIHKLF	GAPOLNDMJKHCFEIB
4	GHCIAFEBJKDPLNOM	PAGIFOEHJNMDCLKB	LNKIFHPBECOGJMAD	IONHAKEBDLMFPCJG
5	HJABDNKMPOLIECFG	JBKOINHDACFGELMP	PLIADGFBKJNCEMOH	MICHBDJLOAGFKPNE
6	EFNPOCDBIGAHMJKL	AHLOGBFNMJDPECKI	AKEHPDJOBCNLGMFI	IEMJPDFNKLCHGBAO

7 Extra Features

In addition to the standard MIPS instructions, you also have to implement push and pop operations in your design using a *Stack*. To achieve this task, the stack memory will be shared with the data memory in the following way. The data memory should start from the minimum address and grow to the increasing memory addresses. On the contrary, the stack memory should start from the maximum address and grow to the decreasing memory addresses. The top of your stack will be held by a stack pointer (**$sp**). Initially **$sp** will hold an address of the highest address of the stack memory. The push and pop operations will be used in the provided assembly code according to the following table and you have to implement them using your MIPS instruction set.

Instruction	Description
push $t0	mem[**$sp**] = **$t0**
push 3($t0)	mem[**$sp**] = mem[**$t0**+3]
pop $t0	**$t0** = mem[**$sp**]

8 Simulation

To simulate your design, an assembly code will be used. Before starting the simulation, you have to convert the given assembly code into MIPS machine code and load the machine code into the instruction memory. The conversion process must be automatic. For example, you can write code in your preferred programming language for this conversion.

9 Report Content

Contents of the report are recommended as follows:

- Section 1: Introduction
- Section 2: Instruction Set
- Section 3: Complete Block diagram of a 4-bit MIPS processor. The block diagram must follow the necessary descriptions.
- Section 4: Block diagrams of the main components.

  - Instruction Memory with PC
  - Data Memory with the Stack
  - Register File
  - Control Unit
- Section 5: Approach to implement the push and pop instructions
- Section 6: ICs used with their count
- Section 7: Discussion

10 Submission Guideline

- Create a folder named “<Lab Group>_<Group ID>_Simulation" and put all the necessary simulation files in this folder.
- Create a second folder named “<Lab Group>_<Group ID>_Necessary_Contents" and put all your codes to convert assembly codes into MIPS machine codes (or any others extra contents that will be necessary to simulate your design).
- Finally create a third folder named “<Lab Group>_<Group ID>_Submission" and put your report along with the previously prepared two folders. Now zip this final folder and upload the zip file to the Moodle submission link (single submission from each group).

11 For Clarification

For any query, you can ask your instructor during the theory or sessional class. Also feel free to email at rahat2975134@gmail.com.