MIPS Design and Simulation

Computer Architecture Sessional

November 15, 2024

1 Problem Description

In this assignment, you will have to design and simulate (in logisim simulation software) a processor that implements the MIPS instruction set architecture. The main components of the processor are Instruction Memory, Data Memory, Register File, Arithmetic Logic Unit (ALU), and Control Unit.

2 Design Specification

⟶ There is a multiplexed Address and Data bus with an 8-bit width.

⟶ Both Data and address are of 8-bits.

⟶ You will have to use 8-bit ALU.

⟶ You will not have to implement ALU; it is shared as a logisim JAR file (attached). You can load this jar file from the option: Project -> Load Library -> JAR Library

⟶ You must design the following temporary registers: `$zero`, `$t0`, `$t1`, `$t2`, `$t3`, `$t4`. All these registers are of 8-bits.

⟶ The Control Unit should be micro-programmed. The control signals associated with the operations should be stored as Control Words in a special memory (you can use a separate ROM for this purpose).

⟶ You must write a program to convert the MIPS assembly code to your MIPS machine code. You can use any programming language for this task.

⟶ Consider efficiency during your design. Marks may be reduced if the design is not efficient enough.

3 Instruction Set Description

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

Table 1: Instruction Set

4 MIPS Instruction Format

MIPS Instructions will be 20 bits in length with the following three formats:

Instruction Type	**Format**				
R-type	Opcode 4-bits	Src Reg 1 4-bits	Src Reg 2 4-bits	Dst Reg 4-bits	Shft Amnt 4-bits
I-type	Opcode 4-bits	Src Reg 1 4-bits	Src Reg 2 4-bits	Address / Immediate 8-bits	
J-type	Opcode 4-bits	Target Jump Address 8-bits		0 4-bits	0 4-bits

Table 2: MIPS Instruction Format

5 Memory Considerations

You need to consider three types of memory:

- Instruction Memory (accessed through program counter, `pc`)
- Data Memory (accessed through address)
- Stack Memory (accessed through stack pointer, `sp`). Sample instructions: `sw $t0, 0($sp)` or `lw $t1, 4($sp)`.

6 Instruction Set Assignment

The opcodes of the instructions will be between 0 and 15 based on the sequence of instruction IDs given below. Sequence `ABCDEFG...` 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	A1	A2	B1	B2	C1	C2
1	GHBICDAEFLJKMOPN	KCPDOALMFNIEBHJG	BGLCEMKDAFJONHIP	NEJKHBFAIMLGPCDO	OEGKMFLIACDPHBJN	ILJCMOKDFPHNEAGB
2	DOLNAHJMPGKFCEBI	CJFHKOENLDBIAMGP	JDKAOFHLMCBEPGNI	EBCDPOLMGJIFKHNA	MPLIDCEOHGAKBJFN	PGEKBCIDJMAONFLH
3	AMHGFDKLPOJIBCNE	BGMFLIDAHOJCKPNE	KNIDHJLAPMGBCOEF	JFPEOLNGMICBAKDH	FCMDPEJAKLIBOHNG	HKILCJEBDFOMPANG
4	NPEOGCJDFHBLAMKI	AKHEPLGMBICFONDJ	DBICPHOJKAMNELGF	PBOELJMCFAIKNDHG	LOAKIJCNBFDHEGMP	MKBONHFAEDCJILGP
5	KGOLAFIEJMDHPBNC	JKHNDOBPCMEGFLIA	LEFPKGIMDAHJNCBO	PMGDKIBNAECLHFOJ	FBHJALMEKOPICNDG	CEMPNIDGKHOBFALJ
6	FBGIDKMAOLCHPENJ	NHEBJGCKOALIMPDF	AMEDOFNJBKCHPLIG	OMIKFCPNDGJHBEAL	DFLGNJEHPBKACIMO	ELBKGCAODIFPMNHJ
7	BOKECIDNFPALMHJG	LCIAFOPKJHDBEGNM	APFMOBLDJICGKENH	HEJIKDBCGNOPFLMA	GKCEBPAIHNJOLDFM	MLDHAKFJICEBGNOP
8	KMFJHLCBIGPEADNO	OBIMGEKCNFDLPAHJ	JCNPLMBIOEFKDHGA	FOCDPEBHNGMIKALJ	APMNEGLIKBDHFOCJ	DLOFKBGHJEPIMANC
9	MNAKDHLJFCEOGPBI	PHLMEJGAFNICBKOD	GKPIFDACBJENOLMH	BDFJMHEOIKPCGNAL	CJOEHFPANMLKIBDG	LOKICDJFABGMHPEN
10	DICJOGEPBLKHANMF	ALFHNEDPKCMOBJGI	EGBOKLCJFAIPMDNH	MHBDPKLOEJACFIGN	PJDOEKMGHICLBNAF	KBCAOINMDLEFGJHP

Table 3: Instruction Set Assignment for Each Group

7 Report Content

Contents of the report are recommended as follows:

- Introduction
- Instruction Set
- Complete Circuit Diagram (of all components)
- How to write and execute a program in your machine
- ICs used with their count
- Contribution of each member
- Discussion

8 Submission Guidelines

- 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_Content` and put your program to convert assembly codes into MIPS machine codes.
- Finally, create a third folder named `<Lab Group>_<Group ID>_Submission` and put your report along with the previously prepared two folders. Zip this folder and upload it to the Moodle submission link (single submission per group).

Submission Deadline: December 6 (Friday), 11 p.m.

For any query, you can mail to **artushar@cse.buet.ac.bd**. Also, you may direct message **ART** on his MS Teams account for a quick response.