Semester Project

Computer System Architecture
MIPS Architecture Design

- Deadline for Project Proposal Submission is 25 December 2015 & Final Deadline is 13 Jan 2016
- It is a group Project. Groups can include at most 3 students.

Project Statement:

Project should include the designing of 16-Bit MIPS processor and modeling of its components using C++. The implementation strategies should be borrowed from most popular 32-bit MIPS architecture. The instruction set should include at-minimum 16 instructions and design the corresponding hardware which should be able to execute the set of instructions properly.
A proper Hardware modules should be designed that would work all together to generate a complete MIPS Hardware Architecture. Hardware Modules include:

- **Instruction Memory**
- **Register File**
- **ALU**
- **Data Memory**

Design should take Assembly instructions as input from user so in order to interpret them, there must be an **Assembler Module** which will convert assembly instruction to the binary instruction.

The Design Specifications

Following assumptions are made in the design of the DSP processor

- The DSP processor has separate data and program memories
- The memories are 256 bits deep so that 8 bits of address may be used to manipulate memory transactions.
- The data memory is 16 bits wide

1. The Instruction Set

The custom Processor supports 2 types of instructions namely

- AGU
- Non-AGU

1.1 Non-AGU type instructions

All the non AGU operations involve the reg-reg operations. Since the operations are reg-reg type so to address a single register we require 3 bit address line. The following diagram specifies the format of the non-AGU type instructions (ADD, MULT, MAC, SHIFT and ALU instructions)

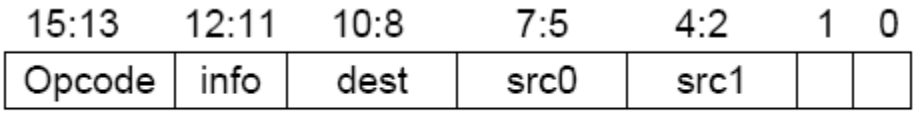

Figure 2 The Non-AGU type instruction format

The shift instruction has a slightly different format since it requires one src register instead of the available two source registers. Owing to the fact that src1 is a 3 bit field the available info field is used to specify the MSBs of the amount of shift. (See examples)
The following table describes the encoding of the op-code and the info fields of the non-AGU type instructions.

Instruction	Opcode (15:13)	Information (12:11)
32 bit ADD	000	00 With saturation 01 Without saturation 10 and 11 Unused
32 bit MULT	001	00 Signed Signed 01 Unsigned Signed 10 Signed Unsigned 11 Unsigned Unsigned
32 bit MAC	010	xx Unused
32 bit SHIFT_L	011	xx Specify the MSBs for the amount of shift (src1)
32 bit SHIFT_A	100	xx Specify the MSBs for the amount of shift (src1)
32 bit ALU opers (xor, and, or not)	101	00 XOR 01 AND 10 OR 11 NOT

Table 2 Opcode and info decoding of non-AGU type instructions

Following couple of examples are for elaboration

ADD_SAT (Add with saturation):

ADD_SAT R0, R1, R2

Will have the following format (the registers taken to be 32 bit wide each)

000	00	000	001	010	0	0

Figure 3 ADD_SAT instruction format

ADD_NSAT (Add without saturation):

ADD_NSAT R0, R1, R2

Will have the following format

000	01	000	001	010	0	0

Figure 4 ADD_NSAT instruction format

MULT_US (multiply unsigned and signed numbers):

MULT_US R0, R1, R2

Will have the following format (Since the multiplication for two 16 bit numbers will result in a 32 bit number the two source registers lower 16 bits only are passed to the multipliers. This consideration must be kept in mind while programming)

001	01	000	001	010	0	0

Figure 5 MULT_US instruction format

SHIFT_L (Logical shift):

SHIFT_L R0, R1, #3

Will have the following format (logical right shift by 3)

011	00	000	001	011	0	0

Figure 6 SHIFT_L(right) instruction format

SHIFT_L (Logical shift):

SHIFT_L R0, R1, #F

Will have the following format (logical right shift by 1)

011	11	000	001	111	0	0

Figure 7 SHIFT_L (left) instruction format

1.2 AGU type instructions

All the AGU operations involve the reg-mem operations (LOAD and STORE) and the PC control operations (BRANCH and RETURN) etc. The load and store instructions require one of the data registers and one of the 4 address registers. Following is the format for the Load and Store instructions.

15:12	11:6	5:2	1	0
Opcode	addr/imm. val	data reg		

Figure 8 The AGU Type instruction format

The load/ store instructions are available in 2 different flavors.

- The 16 bit load from address register.
- The 16 bit load immediate value

In case of load from the address registers only 2 bits (i.e., 11:10) are used to specify the address register in use The RETURN, JUMP and BRANCH instructions require only one address register as a field in their instruction other than the op-code field. Where as the loop instruction requires end address specification as well as the loop count.

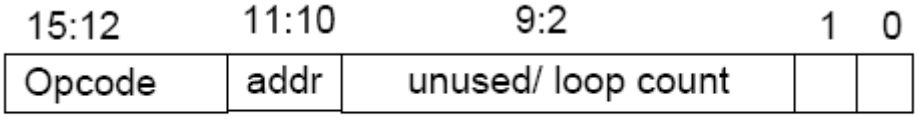

Figure 9 BRANCH, JUMP, LOOP, RETURN instruction Format

Following table specifies the op-code for the AGU instructions

Instruction	Opcode (15:12)	Comments
LOAD	0000	The 16 bit load from address register
	0001	The 16 bit load immediate value
	0010	The 32 bit load from address register
	0011	Not used
STORE	0100	The 16 bit store from address register
	0101	The 16 bit store immediate value
	0110	The 32 bit store from address register
	0111	Not used
JUMP	1000	
BRANCH	1001	
RETURN	1010	
LOOP	1011	

Table 3 Opcode decoding of AGU type instructions