CSE331 / BIL503

COMPUTER ORGANIZATION

FINAL PROJECT: 32-BITS SINGLE CYCLE MIPS PROCESSOR

DUE DATE: 19 JAN 2024, 23:59

In this assignment you will be designing a 32-bit MIPS processor on Verilog. The following instructions should be implemented. The implementation of **MOVE** instruction is a must.

Instruction	Example	Meaning	Type	OPCODE	FUNC. CODE
add	add $1,$2,$3	$1 = $2 + $3	R	000000	000010
subtract	sub $1,$2,$3	$1 = $2 - $3	R	000000	000011
add immediate	addi $1,$2,100	$1 = $2 + 100	I	000010	NA
subtract immediate	subi $1,$2,100	$1 = $2 - 100	I	000011	NA
and	and $1,$2,$3	$1 = $2 & $3	R	000000	000100
or	or $1,$2,$3	$1 = $2 | $3	R	000000	000101
and immediate	andi $1,$2,100	$1 = $2 & 100	I	000100	NA
or immediate	or $1,$2,100	$1 = $2 | 100	I	000101	NA
load word	lw $1,100($2)	$1 = Memory [$2 + 100]	I	001000	NA
store word	sw $1,100($2)	Memory [$2 + 100] = $1	I	010000	NA
load byte	lb $1,100($s2)	$1={24'b0,M[R[rs]+SignExtImm](7:0)}	I	001001	NA
store byte	sb $1,100($2)	M[R[rs]+SignExtImm](7:0)=R[rt](7:0)	I	010001	NA
set on less than	slt $1,$2,$3	if ($2 < $3) $1 = 1; else $1 = 0	R	000000	000111
set on less than immediate	slti $1,$2,100	if ($2 < 100) $1 = 1; else $1 = 0	I	000111	NA
branch on equal	beq $1,$2,100	if ($1 == $2) go to PC + 4 + 100	I	100011	NA
branch on not equal	bne $1,$2,100	if ($1! = $2) go to PC + 4 + 100	I	100111	NA
jump	j 1000	go to address 1000	J	111000	NA
jump and link	jal 1000	$ra = PC + 4; go to address 1000	J	111001	NA
jump register	jr $1	go to address stored in $1	R	000000	001000

Move Instruction:

Instruction	Example	Meaning	Machine Code			
move	**move $rt,$rs**	**$rt = $rs**	100000 (6 bits)	Rs (5 bits)	Rt (5 bits)	0000000000000000 (16 bit)
			PS: 100000 is the opcode of move instruction.			

IMPORTANT! If your design doesn't support MOVE instruction, you get 0.

The machine code of representation of the other instructions are defined below.

R-type:

Opcode (6 bits)	Rs (5 bits)	Rt (5 bits)	Rd (5 bits)	Shamt (5 bits)	Func. Code (6 bits)

I-type:

Opcode (6 bits)	Rs (5 bits)	Rt (5 bits)	Immediate (16 bits)

J-type:

Opcode (6 bits)	Address (26 bits)

- The memory addresses will be 18-bits. Which means your data memory is 256 KB.
- Instruction memory should hold up to 1024 instructions where each instruction is 32-bits. Thus, the least significant 10 bits of the jump addresses will be enough. But you should fill the rest of the address block with zeros. For instance, jump instruction's address block (in machine code level) should be 26 bits even though you only use the least significant 10 bits.
- The information (the names, inputs, outputs etc.) about the some of the verilog modules will be announced. **You should stick into that format. If you don't, you get 0, even though your implementation works.**
- **If you don't implement MOVE instruction, you get 0.**
- **If your design is not working, you will get at most 30 pts.**
- You can use the ALU you designed for Project 2 (but you will need to upgrade it).
- Don't forget byte-access (for lb and sb) and sign extend (for immediate instructions).
- Despite there is a "Shamt" block in the machine code representation of R-type instructions, you will not implement shift instructions. Therefore, Shamt will always be "00000".
- This document doesn't describe the project fully. The details will be shared in the Problem Session on 25th December, 14:00.
- You will be using structural verilog only, except a few modules (which will be shared).
- You will be reading instructions from instruction memory (which is a file in this project) and then write the results to registers or memory (which are also files). More information about file operations will be shared in the Problem Session.
- You will also write a testbench, the details will be shared.
- Since this is a detailed project, new requirements could be shared. You are responsible of following the announcements made on Teams page.

Bonus Part: Any successful hardware implementation of the project grants you **extra 25 points**. Such an implementation requires a face-to-face demo.

PS: The hardware will be provided for the students who wish to do the bonus part. Thus, you need to contact with TA.

PS: The weight of this project is higher than the previous ones.