

Computer-Aided VLSI System Design, Fall 2011

Verilog Homework 4 – Testbench Writing

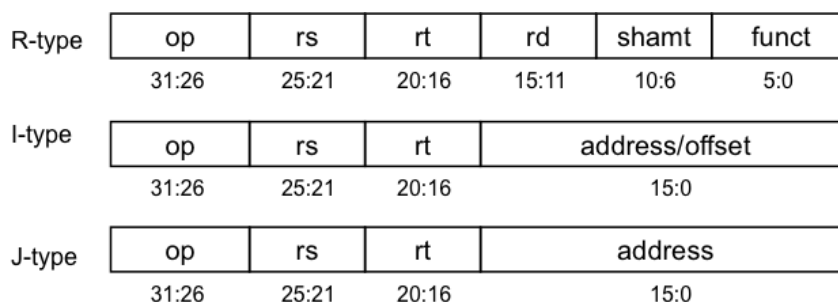
I. Objective

Please write the testbench to debug a single cycle MIPS CPU. There are **some errors** in the design. You need to write the testbench to check all instruction sets, and then correct these errors. These errors include the **function errors** and **coding errors**.

II. Description of MIPS CPU

A. Supported instructions:

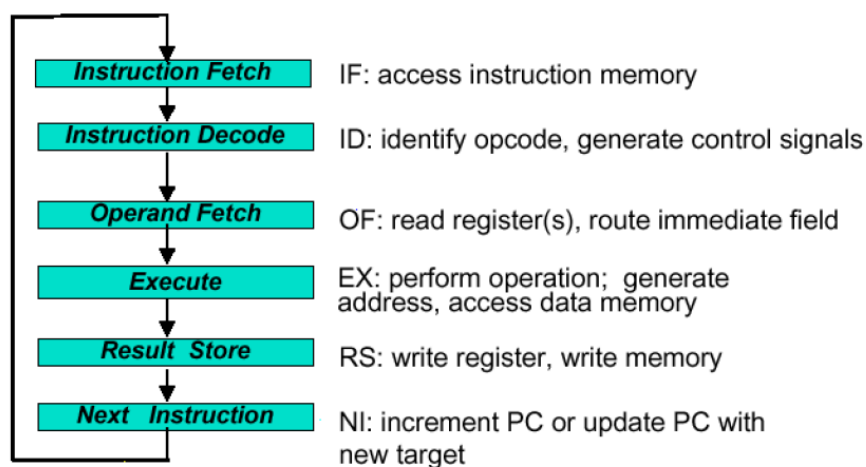
MIPS instruction	Example	Meaning	Format (opcode)	
Add: add	add \$1,\$2,\$3	\$1 = \$2 + \$3 (rd=\$1, rs=\$2, rt=\$3)	R-type	00_0000
Subtract: sub	sub \$1,\$2,\$3	\$1 = \$2 - \$3	R-type	00_0000
And: and	and \$1,\$2,\$3	\$1 = \$2 & \$3	R-type	00_0000
OR: or	or \$1,\$2,\$3	\$1 = \$2 \$3	R-type	00_0000
Set less than: slt	slt \$1,\$2,\$3	if (\$2<\$3) \$1=1; else \$1=0;	R-type	00_0000
Load word: lw	lw \$1,100(\$2)	\$1 = Mem[\$2+100] (rs=\$2, rt=\$1, offset=100)	I-type	10_0011
Store word: sw	sw \$1,100(\$2)	Mem[\$2+100] = \$1	I-type	10_1011
Branch on equal: beq	beq \$1,\$2,L	if (\$2==\$3) go to L; (address = L/4)	I-type	00_0100
Jump: j	j L	Go to L; (address = L/4)	J-type	00_0010
Jump and link: jal (for procedure call)	jal L	\$ra = PC+4, go to L ; (\$ra=\$31)	J-type	00_0011
Jump register: jr (for procedure return)	jr \$ra	go to \$ra (rd=\$ra)	R-type	00_0000



B. ALU OP Code

Instruction opcode	ALUOp	Instruction operation	Funct field	Desired ALU action
LW	00	load word	xxxxxx	add
SW		store word	xxxxxx	add
Branch equal	01	branch equal	xxxxxx	subtract
R-type	10	add	100000	add
R-type		subtract	100010	subtract
R-type		AND	100100	and
R-type		OR	100101	or
R-type		set on less than	101010	set on less than

C. Typical Instruction Execution



Note that each step does not necessarily correspond to a clock cycle. These only describe the basic flow of instruction execution. The details vary with instruction type.

Instruction class	Functional units used by the instruction class				
R-type	Instruction fetch	Register access	ALU	Register access	
Load word	Instruction fetch	Register access	ALU	Memory access	Register access
Store word	Instruction fetch	Register access	ALU	Memory access	
Branch	Instruction fetch	Register access	ALU		
Jump	Instruction fetch				

1. Instruction fetching

- The CPU is always in an infinite loop, fetching instructions from memory and executing them.
- The program counter or PC register holds the address of the current instruction.
- MIPS instructions are each four bytes long, so the PC should be incremented by four to read the next instruction in sequence.

2. Instruction Decode

- The Instruction Decode (ID) step reads the source register from the register file.

3. Execute

- The third step, Execute (EX), computes the effective memory address from the source register and the instruction's constant field.

4. Memory

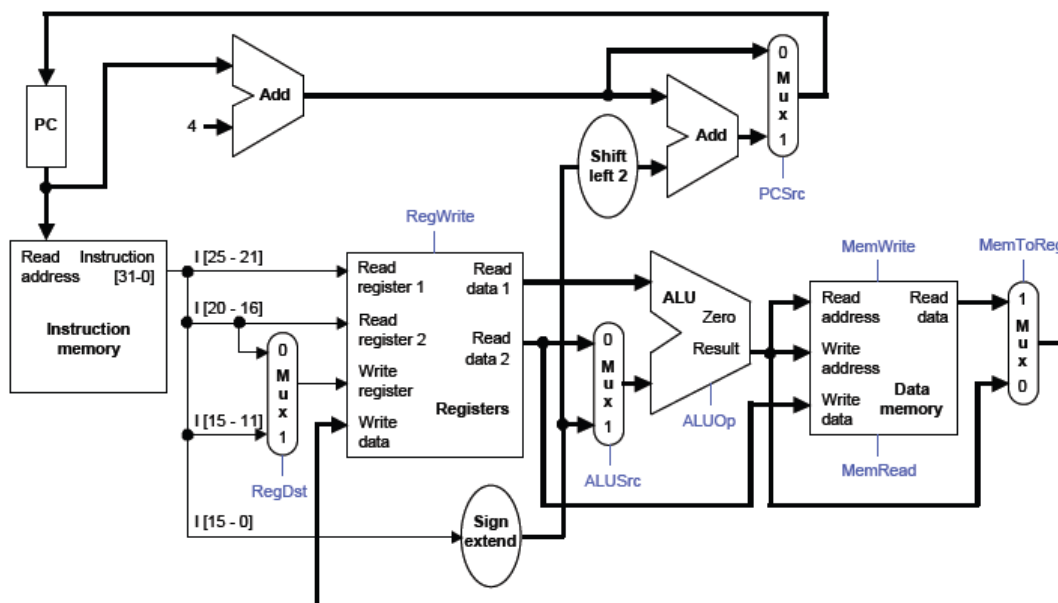
- The Memory (MEM) step involves reading the data memory, from the address computed by the ALU.

5. Write Back

- Finally, in the Write Back (WB) step, the memory value is stored into the destination register.

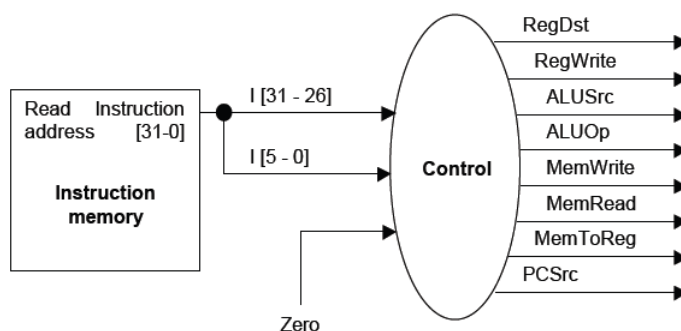
D. Architecture of MIPS**1. Data Path**

- Instruction memory, registers file (\$0, \$1...\$31), ALU, ALU control, data memory, program counter (PC), sign-extend unit.



2. Control Unit

- The control unit needs 13 bits of inputs.
 - Six bits make up the opcode of instruction.
 - Six bits come from the func field of instruction.
 - It also needs the Zero output of the ALU.
- The control unit generates 10 bits of output, corresponding to the signals mentioned on the previous page.



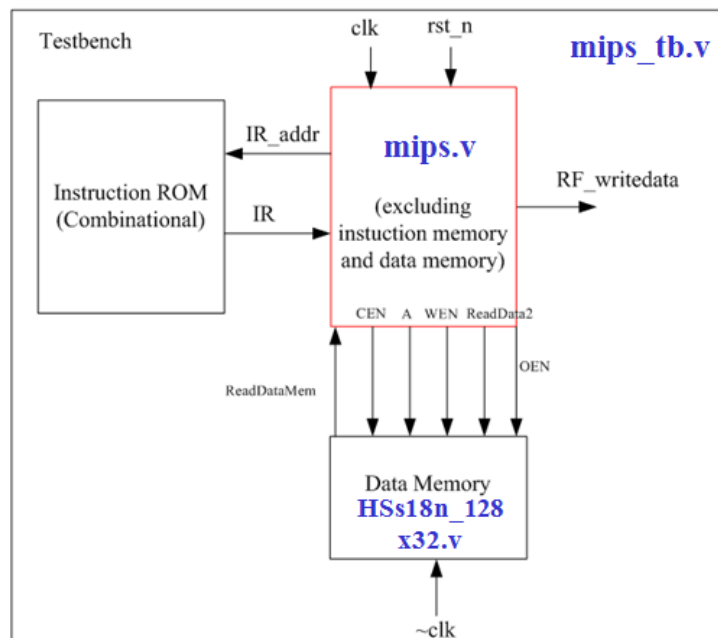
III. File List & Block Diagram & I/O Definition

A. File list of the homework package:

mips.v	A single cycle MIPS CPU with some errors.
mips_tb.v	Empty file for you to write the testbench.
HSs18n_128x32.v	A 128X32 high-speed single-port synchronous SRAM.

B. Block Diagram

Testbench includes the **mips**, **data memory** and **instruction ROM**.
Instruction ROM contains the testing instructions.



E. File I/O definition:

clk	Posedge clock signal.
rst_n	Active low asynchronous reset.
IR_addr	Instruction address
IR	Instruction
RF_writedata	The data to be written into the register file. Used for testing your circuit.
ReadDataMem	MIPS- Read data from memory. SRAM- Data outputs.
CEN	SRAM- Chip enable. 0 when you read/write data.
WEN	SRAM- Write enable. 0 when you write data into SRAM
A	SRAM- Address
ReadData2	MIPS- Write data to memory. SRAM-Data inputs.
OEN	SRAM- Read Enable. 0 when you read data from SRAM

IV. Testbench Requirements

- A. Instruction memory and data memory are included in the testbench.
- B. Initialize the instruction memory and the data memory.
- C. Reset your circuit. Besides, the register can be reset to a value what you want for verification.
- D. **Execute all instructions:**

You can design the test pattern for debugging. Besides, you need to print the debugging information. For example:

MIPS Test1 Start:

add \$1, \$2, \$3: \$2=6, \$3=2, \$1=8 → correct

sub \$1, \$2, \$3: \$2=6, \$3=2, \$1=6 → error

....

....

- E. **Run the specific program:**

You need to run Fibonacci number program to test the MIPS.

$$F_n = F_{n-1} + F_{n-2}$$

$$F_0 = 0 \quad F_1 = 1$$

Fibonacci number is 0,1,1,2,3,5,8,13,21....

Also, need to print the total program and debugging information of Fibonacci number program (n=7). Finally, print the correct Fibonacci number. For example:

MIPS Test2 Start: (Fibonacci number n=7)

beq \$3, \$0, 32

....

....

Fibonacci number is 0,1,1,2,3,5,8,13

- F. If your function is correct, please print the message is shown as below:

Congratulations!! Your design has passed all tests ^_^

V. Grading Policy

1. (60%) mips.v correctness
report: Please **show the errors** of MIPS and **correct these errors**.
2. (40%) mips_tb.v correctness: requirement D (20%) & requirement E (20%)

VI. Submission Requirements

File submission: **1.mips.v (no errors), 2.mips_tb.v, 3.report file**

Please submit your design in one .zip or .rar file with the naming convention:

StudentID_HW4.zip/.rar (e.g., R00943001_HW4.zip).

VII. Submission Deadline (Deadline: 12/2 Friday)

Please submit your **.zip** or **.rar** files online using ftp.

VIII. Reference Book

David A. Patterson and John L. Hennessy, "Computer Organization & Design: The Hardware/Software Interface. 3rd edition"

Creator:

1st Edition: En-Jui Chang, 2010

2nd Edition: En-Jui Chang, 2011