Computer Architecture Experiment

Lab 1: Warmup

Outline

- Experiment Purpose
- Experiment Task
- Basic Principle
- Operating Procedures
- Precaution

Experiment Purpose

- Warmup with what you have learned in the course of Computer Organization
- Try to use the ISE environment skillfully
- Read the user guide of Sparten-3E board, especially the part of LED displayer.

The 9 MIPS instructions（ORG lab)

	Instruction bit number					
	31..26	25..21	20..16	15..11	10..6	5..0
add	000000	rs	rt	rd	00000	100000
sub	000000	rs	rt	rd	00000	100010
and	000000	rs	rt	rd	00000	100100
or	000000	rs	rt	rd	00000	100101
slt	000000	rs	rt	rd	00000	101010
lw	100011	rs	rt	immediate		
sw	101011	rs	rt	immediate		
beq	000100	rs	rt	immediate		
j	000010	address				

Experiment Task

- Update your Verilog code in Computer Organization to implement and test the single cycle CPU on SP3E board.
- Extend your CPU (in Org) to support 15 instructions and run on SP3E.

Lab steps:

- Get warmup with Verilog, ISE and SP3E.
- Implement to show results on LCD instead of segment or Leds. (LCD code can be found in course website.)
- Update your CPU to implement 15 instructions. (Draw the logical graph first)
- Design testcode testing each instruction.
- Run the test code and show the results.
  - Step-by-step mode（must)
  - Execute （ selective)
- Displaying results
  - PC, instruction code, register values (32)

Prepare for checking

- Do understand what you have implemented.
  - Show your testcode (both in assembler code and machine code when you present your result.
  - Prepare to answer questions on your results (Verilog code, logical graph, UCF…).

Report Submission

- All reports should be submitted to course website.
- Submision file should be named as: "StID_Name_lab1.rar", including StID_name_Lab1.doc (report) and all the .v files, ucf file, .coe file, .bit file, and test code.
- Report should be rewritten according to the report template.

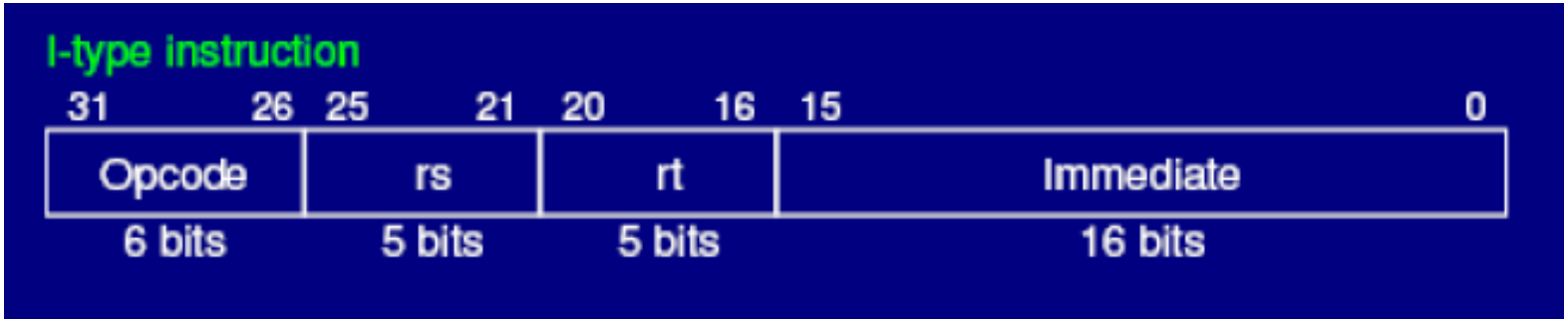
15 common used MIPS instructions

MIPS Instructions								
Bit #	[31..26]	[25..21]	[20..16]	[15..11]	[10..06]	[05..00]	Operations	
R-type	op	rs	rt	rd	sa	func		
add	000000	rs	rt	rd	00000	100000	rd <-- rs + rt;	PC <-- PC + 4
sub	000000	rs	rt	rd	00000	100010	rd <-- rs - rt;	PC <-- PC + 4
and	000000	rs	rt	rd	00000	100100	rd <-- rs & rt;	PC <-- PC + 4
or	000000	rs	rt	rd	00000	100101	rd <-- rs | rt;	PC <-- PC + 4
sll	000000	00000	rt	rd	sa	000000	rd <-- rt << sa;	PC <-- PC + 4
srl	000000	00000	rt	rd	sa	000010	rd <-- rt >> sa (logical);	PC <-- PC + 4
sra	000000	00000	rt	rd	sa	000011	rd <-- rt >> sa (arithmetic);	PC <-- PC + 4
I-type	op	rs	rt	immediate				
addi	001000	rs	rt	immediate			rt <-- rs + (sign_extend)immediate;	PC <-- PC + 4
andi	001100	rs	rt	immediate			rt <-- rs & (zero_extend)immediate;	PC <-- PC + 4
ori	001101	rs	rt	immediate			rt <-- rs | (zero_extend)immediate;	PC <-- PC + 4
lw	100011	rs	rt	immediate			rt <-- memory[rs + (sign_extend)immediate];	PC <-- PC + 4
sw	101011	rs	rt	immediate			memory[rs + (sign_extend)immediate] <-- rt;	PC <-- PC + 4
beq	000100	rs	rt	immediate			if (rs == rt) PC <-- PC + 4 + (sign_extend)immediate<<2; else	PC <-- PC + 4
bne	000101	rs	rt	immediate			if (rs != rt) PC <-- PC + 4 + (sign_extend)immediate<<2; else	PC <-- PC + 4
J-type	op	address						
j	000010	address					PC <-- (PC+4)[31..28],address<<2	

Branch instructions

Example instruction	Instruction name	Meaning
J name	Jump	PC←name; ((PC+4)-2^{25}) ≤ name < ((PC+4)+2^{25})
JAL name	**Jump and link**	**Regs[31]←PC+4; PC←name; ((PC+4)-2^{25}) ≤ name < ((PC+4)+2^{25})**
JALR R2	Jump and link register	Regs[31]←PC+4; PC←Regs[R2]
JR R3		PC ← Regs[R3]
BEQ R4, R5, name	Branch equal	If (Regs[R4]==Regs[R5]) PC←name; ((PC+4)-2^{25}) ≤ name < ((PC+4)+2^{25})
BNE R4, R5, name	**Branch not equal**	**If (Regs[R4] ≠Regs[R5]) PC←name; ((PC+4)-2^{25}) ≤ name < ((PC+4)+2^{25})**

Instruction format



- Thanks!