CSE/EEE 230: Computer Organization and Assembly Programming

Project 1

Date assigned: September 24th, 2012

Due date: 4.30 pm, October 8th, 2012 (at the start of the lecture)

1. [20 points] Modify demo program and run it.

i. Follow the demo instruction to run dot product program.

ii. ***Task 1*** Modify the program to calculate dot product of the following two vectors:
Vector A = [6,5,2,3,4,1] = [0x6,0x5,0x2,0x3,0x4,0x1]
Vector B = [0,2,3,7,8,10] = [0x0,0x2,0x3,0x7,0x8,0xA]

2. [40 points] Use SLL/SLLV to achieve 'multiply'.

SLL/SLLV shifts register value to the left. When shifting 1 bit, the value is multiplied by 2. When shifting 2 bits, the value is multiplied by 4 and so on. This instruction is always used to multiply a value by x_{th} power of 2.

Task 2 Use SLL/SLLV instead of MULT/MULTU to calculate dot product of the following two vectors:
Vector A = [6,5,2,3,4,1] = [0x6,0x5,0x2,0x3,0x4,0x1]
Vector B = [2,4,8,16,32,64] = [0x2,0x4,0x8,0x10,0x20,0x40]

Reference:

SLL -- *Shift left logical*

Description:	Shifts a register value left by the shift amount listed in the instruction and places the result in a third register. Zeroes are shifted in.
Operation:	$d = $t << h; advance_pc (4);
Syntax:	sll $d, $t, h
Encoding:	`0000 00ss ssst tttt dddd dhhh hh00 0000`

SLLV -- *Shift left logical variable*

Description:	Shifts a register value left by the value in a second register and places the result in a third register. Zeroes are shifted in.
Operation:	$d = $t << $s; advance_pc (4);
Syntax:	sllv $d, $t, $s
Encoding:	`0000 00ss ssst tttt dddd d--- --00 0100`

3. [40 points] Use ADD/ADDI/ADDIU/ADDU to achieve ‘multiply’.

Multiplier is not always power of 2. In this case, ‘multiply’ can be achieved by regular addition. When you multiply X by Y, the result can be achieved by adding X Y times.

Task 3 Use ADD/ADDI/ADDIU/ADDU instead of MULT/MULTU to calculate dot product of the following two vectors:

Vector A = [6,5,2,3,4,1] = [0x6,0x5,0x2,0x3,0x4,0x1]

Vector B = [0,2,3,7,8,10] = [0x0,0x2,0x3,0x7,0x8,0xA]

Reference:

ADD – *Add (with overflow)*

Description:	Adds two registers and stores the result in a register
Operation:	$d = $s + $t; advance_pc (4);
Syntax:	add $d, $s, $t
Encoding:	`0000 00ss ssst tttt dddd d000 0010 0000`

ADDI -- *Add immediate (with overflow)*

Description:	Adds a register and a sign-extended immediate value and stores the result in a register
Operation:	$t = $s + imm; advance_pc (4);
Syntax:	addi $t, $s, imm
Encoding:	`0010 00ss ssst tttt iiii iiii iiii iiii`

ADDIU -- *Add immediate unsigned (no overflow)*

Description:	Adds a register and a sign-extended immediate value and stores the result in a register
Operation:	$t = $s + imm; advance_pc (4);
Syntax:	addiu $t, $s, imm
Encoding:	`0010 01ss ssst tttt iiii iiii iiii iiii`

ADDU -- *Add unsigned (no overflow)*

Description:	Adds two registers and stores the result in a register
Operation:	$d = $s + $t; advance_pc (4);
Syntax:	addu $d, $s, $t
Encoding:	`0000 00ss ssst tttt dddd d000 0010 0001`

Deliverables:

1 Your code should be properly commented.

2 Name your code as ***Task_#_Your_Name.asm*** and upload them to BlackBoard.