University of California, Berkeley
College of Engineering
Computer Science Division — EECS

Spring 2001 John Kubiatowicz

Homework Quiz (HW #4)

March 4, 2001
CS152 Computer Architecture and Engineering

This quiz covers one of the problems from homework #4.
Good Luck!

Your Name:	
SID Number:	
Discussion Section:	

Total:	

The BSET instruction

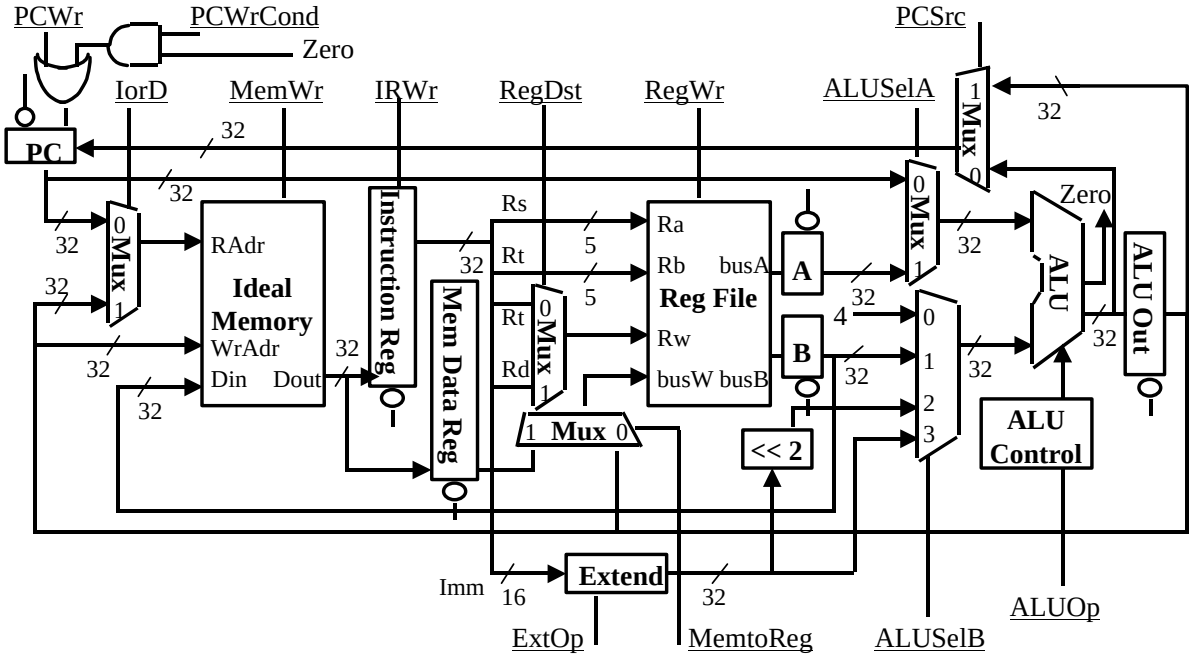

Figure 1: A multicycle data path

Figure 1 shows the multicycle datapath from the book. In your homework, you implemented the **`bcp`** instruction, which copied a block of words from one location in memory to another. In this problem, we will implement something much simpler, namely the block set instruction:

```
bset  $t1,$t2,$t3
        ⇒ {$t1⇒ mem[t3], mem[t3+1] … mem[t3+t2]}
```

This instruction takes the value in register $t1, and stores it back to a block of words of memory of size $t2 words, starting at the address in $t3. The coding of this instruction is as follows: $t1 is in the RT field, $t2 is in the RS field, $t3 is in the RD field.

Problem 1a: What changes are required to the *datapath* for this instruction? Assume that you can use a 32-bit counter with separate **load** and **decrement** signals. Only redraw/describe the affected parts of the datapath. Do not change the register file. *Big hint: you should only need 1 new mux (near register file), 1 new counter, and one new input to an existing mux. Model after store word.*

Field Name	Values For Field	Function of Field
ALU	`Add`	ALU Adds
	`Sub`	ALU subtracts
	`Func`	ALU does function code (Inst[5:0])
	`Or`	ALU does logical OR
SRC1	`PC`	PC ⇒ 1st ALU input
	`rs`	R[rs] ⇒1st ALU input
SRC2	`4`	4 ⇒ 2nd ALU input
	`rt`	R[rt] ⇒ 2nd ALU input
	`Extend`	sign ext imm16 (Inst[15:0]) ⇒ 2nd ALU input
	`Extend0`	zero ext imm16 (Inst[15:0]) ⇒ 2nd ALU input
	`ExtShft`	2nd ALU input = sign extended imm16 << 2
ALU Dest	`rd-ALU`	ALUout ⇒ R[rd]
	`rt-ALU`	ALUout ⇒ R[rt]
	`rt-Mem`	Mem input ⇒ R[rt]
Memory	`Read-PC`	Read Memory using the PC for the address
	`Read-ALU`	Read Memory using the ALUout register for the address
	`Write-ALU`	Write Memory using the ALUout register for the address
MemReg	`IR`	Mem input ⇒ IR
PC Write	`ALU`	ALU value ⇒ PCibm
	`ALUoutCond`	If ALU Zero is true, then ALUout ⇒ PC
Sequence	`Seq`	Go to next sequential microinstruction
	`Fetch`	Go to the first microinstruction
	`Dispatch`	Dispatch using ROM

Label	ALU	SRC1	SRC2	ALUDest	Memory	MemReg	PCWrite	Sequence
Fetch	Add	PC	4		ReadPC	IR	ALU	Seq
Dispatch	Add	PC	ExtShft					Dispatch
SW	add	rs	rt					Seq
					Write-ALU			Fetch

Problem 1b: Above, we show the microcode assembly-language and the microcode for the store word(SW) instruction. What changes are needed to the microcode assembly language to support the `bset` instruction? *Hint: something will be added to the Sequence field!*

Problem 1c: Write complete microcode for `bset`, including the *fetch* and *dispatch* cycles. Be careful not to make changes that would interfere with the other instructions. You shouldn't need more than 5 or 6 total instructions (including fetch and dispatch):

Filename:	homequiz_4.doc
Directory:	C:\usr\homes\kubitron\courses\cs152-S01\quizzes
Template:	C:\Documents and Settings\kubitron.COMP-SCI\Application Data\Microsoft\Templates\Normal.dot
Title:	University of California, Berkeley
Subject:	
Author:	vwen
Keywords:	
Comments:	
Creation Date:	3/6/2001 7:53 AM
Change Number:	9
Last Saved On:	3/29/2001 3:49 PM
Last Saved By:	John Kubiatowicz
Total Editing Time:	226 Minutes
Last Printed On:	3/29/2001 3:50 PM
As of Last Complete Printing	
Number of Pages:	3
Number of Words:	441 (approx.)
Number of Characters:	2,519 (approx.)