Lab 12

Single Cycle MIPS Datapath

(Data memory and Branch)

* Pre-Lab

1. Today’s Task

In this Lab you will add data memory and circuitry for branch instruction. The output for this block is the output of third mux which could be; either data read from data memory (lw), or, the output of ALU (R-type result or sw address).

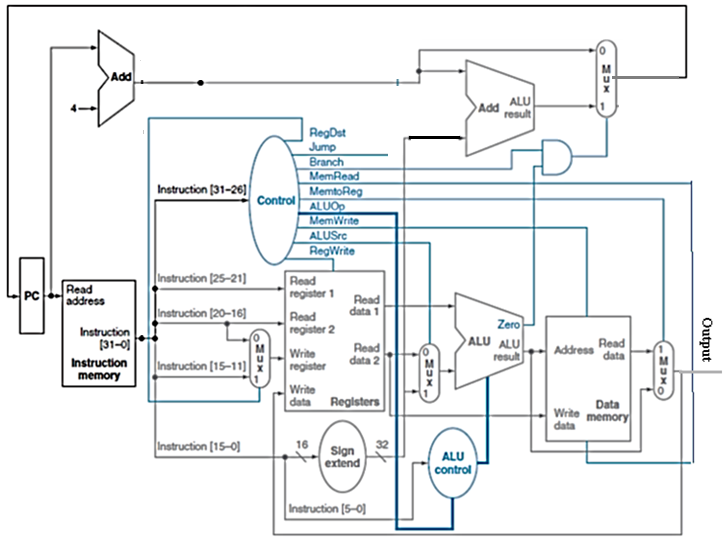
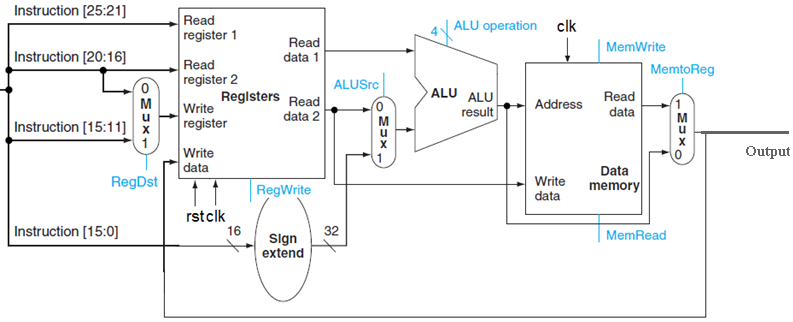


Figure 12.1

* 1. Part1

In Part1 add data memory (20 locations, 32-bit wide) to previous block (up to ALU). Connect the output of data memory through mux to regfile. Data memory is similar to instruction memory except Data is only read when MemRead signal is asserted (value1). Data is written only at positive edge of clock when MemWrite is asserted.

Figure 12.2

* + 1. **Test Bench for Part 1**

Write the following instruction in the instruction memory.

|  |  |  |  |
| --- | --- | --- | --- |
| Address in memory | Instruction | With register Numbers | Corresponding  Hexadecimal Code |
| 0 | lw $t1,4($t2) | lw $9,4($10) | 0x8d490004 |
| 1 | add $t3,$t2,$t1 | add $11,$10,$9 | 0x01495820 |
| 3 | sw $t3,5($t2) | sw $11,5($10) | 0xad4b0005 |

Table 12.1

* **lw $t1,4($t2)**(lw $9,4($10))

This instruction will read data from memory location “4 + value in $10”.

$9=DataMemory[4+$10]

For example if $t2($10)=8, this instruction will read data from memory location 4+8=12, and write the result in register $t1($9). Assign values of your choice to $t2, and memory location ‘4+$t2’.

**Output= DataMemory[4+$10]**

* **add $t3,$t2,$t1**(add $11,$10,$9)

It will add register $9 and $10 and write result in $12.

**Output= $10+$9**

* **sw $t3,5($t2)**(sw $11,5($10))

It will write the value of $11 to memory location “5 + value of $t2”. Assign value of your choice to $12.

**Output= 5+$10**

* 1. Part2(Branch)

Mips beq $rs,$rt,offset effects PC(program counter). On the basis of the comparison result of rs and rt it is decided whether to normally increment the PC, or, add the offset to it. If rs≠rt, PC=PC+1 (normal operation), otherwise PC=(PC+1)+offset.

If you will look the tables regarding Control Unit and ALU, you will see that beq instruction uses ‘Subtraction’ operation.

The circuit given below adds offset(instruction[15:0]) to PC+1. The mux selects between PC+1 and PC+1+offset, if Branch=1 and Zero=1(which means both operands are equal)

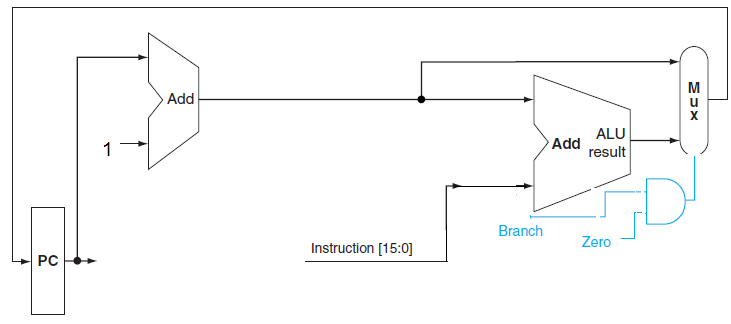


Figure 12.3

Offset is Instruction[15:0], and it is the relative address of instruction with the label from the beq instruction. For example for the following code

*Label1:* add $t1,$t1,$t0 #$t1=$t1+1

beq $t1,$t2,Label2

j Label1

*Label2:* add $t1,$t1,$t2

**Code1**

The value offset(distance between beq and Label2) is 1. It isn’t ‘2’ because PC=(PC+1)+offset will make the final change of +2

For the following code

Label: add $t1,$t1,$t0 #$t1=$t1+1

bne $t1,$t2,Label

**Code2**

The value of offset is -2, because PC=(PC+1)+offset will make the final change of -1

* + 1. **Test Bench for Part 2**

Similar to part1 but this time you do not need to give the control signals since appropriate control signals will be given by Control Unit

**#Initializing Register File**

#Move Value 0 in RegFile[2] (initializing)

#Move Value 1 in RegFile[3] (used for incrementing $2)

#Initialize RegFile[4] with any value of your choice, let’s say 456

**#Initializing Data Memory**

#Initialize DataMem[0] ----DataMem[10] with value stored in RegFile[4] which in this case is 456

The Following code has a loop that will run 10 times

loop:

lw $1,0($2) #$1=DataMem[$2]

add $2,$2,$3 #$2=$2+1

beq $1,$4,loop #Compare $1 with RegFile[4]

The Hex Code for instruction memory is

32’h8c410000

32’h00431020

32’h1024fffd

* In-Lab
* Write Verilog HDL decription of
  + Data memory
  + Branch Datapath

And add to previous blocks

* Post-Lab

Submit the report of complete single-cycle implemetation