CSEE W3827

Fundamentals of Computer Systems

Homework Assignment 4

Solutions

Profs. Stephen A. Edwards & Martha Kim

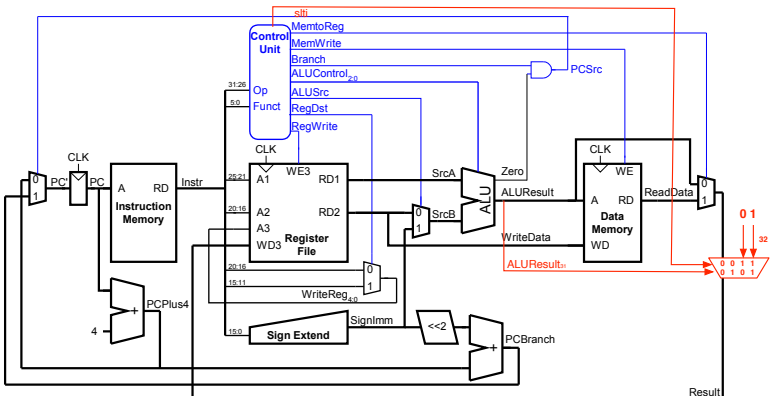
Columbia University

Due December 4, 2012 at 10:10 AM

Write your name **and UNI** on your solutions

Show your work for each problem; we are more interested in how you get the answer than whether you get the right answer.

1. (25 pts.) Extend the single-cycle MIPS processor to support `slti` (i-format, opcode=001011).



Inst.	OP	RegWrite	RegDst	ALUSrc	Branch	MemWrite	MemToReg	ALUOp	slti
R-type	000000	1	1	0	0	0	0	10	0
lw	100011	1	0	1	0	0	1	00	0
sw	101011	0	-	1	0	1	-	00	0
beq	000100	0	-	0	1	0	-	01	0
slti	001011	1	0	1	0	0	-	01	1

2. (15 pts.) A processor that does not support the `slti` instruction, can emulate the behavior as follows:

```
subi $d, $s, imm
srl  $d, $d, 31
```

Imagine two single-cycle processors, P_{base} which does not support `slti`, and P_{slti} which does. Consider also two versions of an application: A_{slti} which is dynamically 10% `slti` instructions, and A_{base} which is the same application with all of the `slti`'s replaced by the pair of instructions above. Assuming both processors run at the same clock frequency, rank the following combinations of software and hardware, from fastest to slowest, explaining your reasoning.

- A_{base} on P_{base}
- A_{base} on P_{slti}
- A_{slti} on P_{slti}

Because A_{base} has 10% more instructions than A_{slti}, it will take 10% longer to complete, regardless of the processor it is on. Thus, A_{slti} on P_{slti} is the fastest, followed by a two way time of A_{base} on either processor.

3. (25 pts.) Consider the tree sum code below.

```
tree_sum:
    bnez  $a0, tree_sum_recurse # non-pseudo: bne $a0, $zero, tree_sum_recurse
    li    $v0, 0
    jr    $ra
tree_sum_recurse:
    addi  $sp, $sp, -12
    sw   $ra, 0($sp)
    sw    $s0, 4($sp)
    sw    $s1, 8($sp)
    move  $s0, $a0              # non-pseudo: addi $s0, $a0, 0
    lw    $s1, 0($s0)
    lw    $a0, 4($s0)
    jal   tree_sum
    add   $s1, $s1, $v0
    lw    $a0, 8($s0)
    jal   tree_sum
    add   $v0, $s1, $v0
    lw    $ra, 0($sp)
    lw    $s0, 4($sp)
    lw    $s1, 8($sp)
    addi  $sp, $sp, 12
    jr    $ra
```

(a) What is the total number of dynamic instructions, when `tree_sum` is called on an N-node tree?

When run on an N-node tree, this function will recurse N times and hit the base case N+1 times. In the base case, 3 instructions are executed (bnez, li, jr). In the recursive case, 1 + 17 instructions are (bnez, addi, sw, sw, ... lw, lw, addi, jr). So the total number of instructions is $(N+1)\times 3+(N\times 18)=3N+3+18N=21N+3$.

(b) What is the dynamic ratio of control transfer, memory (i.e, loads and stores), and arithmetic instructions (including `li` and `move`) on an N-node tree?

Usinc C, M, and A to represent the three classes, we can rewrite the above expression in terms of the types of instructions:
$(N+1)\times(2C+A)+(N\times(9M+5A+4C))=2CN+2C+NA+A+9MN+5AN+4CN=$
$N\times(2C+A+9M+5A+4C)+2C+A=N\times(6C+9M+6A)+2C+A$

(c) Assuming a very large tree (millions of nodes) what is the rough overall instruction mix?

$\frac{6}{21}\times C+\frac{9}{21}\times M+\frac{6}{21}\times A=28\%$ each arithmetic and control transfer, plus 45% memory.

(d) Consider a single cycle CPU running at 500MHz and a multicycle CPU (5 cycles for memory ops, 3 for control transfers, 4 for arithmetic) running at 2GHz, which will compute tree sums faster?

$CPI_{MC}=.28\times 4+.28\times 3+.45\times 5=4.49$ The multicycle clock frequency is 4x the single cycle, but the former reqires >4x as many cycles per instruction, so the single cycle processor will complete the computation first.

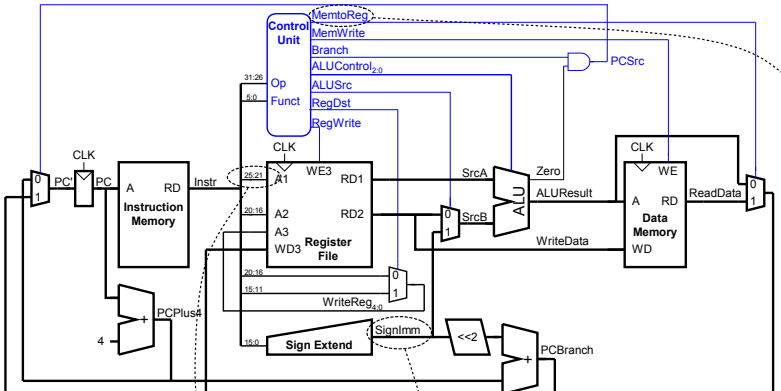
4. (35 pts.) Fill in the values found on the indicated wires for the first 10 cycles when the `tree_sum` code above is executed on the following tree:

```
tiny:       .word 1, tiny1, tiny2
tiny1:      .word 2, 0, 0
tiny2:      .word 3, 0, 0
```

Assume that the processor supports `addi` *and* `jal`*; If any values are undefined given the information provided, simply mark as "undefined".*

First, we figure out what ten instructions are being executed:

```
bnez  $a0, tree_sum_recurse
addi  $sp, $sp, -12
sw    $ra, 0($sp)
sw    $s0, 4($sp)
sw    $s1, 8($sp)
move  $s0, $a0
lw    $s1, 0($s0)
lw    $a0, 4($s0)
jal   tree_sum
bnez  $a0, tree_sum_recurse
```



MemtoReg
x
0
x
x
x
0
1
1
x
x

Instr25:21	SignImm
$zero/0 or $a0/4	0x00000002 / 2
$sp/29	0xfffffff4 / -12
$sp/29	0x00000000 / 0
$sp/29	0x00000004 / 4
$sp/29	0x00000008 / 8
$a0/4	0x00000000 / 0
$s0/16	0x00000000 / 0
$s0/16	0x00000004 / 4
undefined	undefined
$zero/0 or $a0/4	0x00000002 / 2