

Comp 120 Computer Organization
Spring 2005

Solutions to Problem Set #7

Problem 1. “Life Without LUI” [25 points]

(A) `addi $rd, $0, constant`
`sll $rd, $t0, 16`

(B) `0xfeedface` is loaded to the `$t0` after the execution of the instruction labeled “skip”.

Since the program stores the next instruction address to `$31` before jumping to “skip”, the content originally in `$31` will be overwritten. A correct way to use this approach to load large constant is to move `$31`'s content to some other register that will not be written during execution of “skip”, and move back to `$31` after returning from “skip”.

To load one big constant, the standard approach needs two instructions and 2 words in memory to store the instructions; the approach in (A) need three instructions and 3 words in memory; the new approach here both takes two instructions and 3 words in memory. If we only need to load one constant, then standard approach is best.

However, if we load n constants, we need $2n$ instructions in standard way, $3n$ instructions in (A), but only $(1+n)$ instructions with this new approach. That is, we can store the constants consecutively immediately after the `jal` instruction, and load consecutively in “skip”. For large n , the new technique is best.

(C) The i -th constant is at address $(\$gp) - 4i$. Suppose the compiler knows this shift amount $-4i$.

```
lw $t1, -4i($gp)
```

This new approach here takes one instructions to load one constant, but the proposed convention requires that constants come before global and static variables. Besides, the shift amount is in $[-32768, -4]$, so we can at most load 8192 constants.

(D) Load the upper part to one register then shift left 16 bits. Load the lower part to another register, and add to the first register.

```
addi $t0, $0, 16;
```

```

    lw $t1, 0xupper
loop: addi $t1, $t1, $t1
    subi $t0, $t0, 1
    bne $t0, $0, loop
    lw $t0, 0xlower
    add $t1, $t1, $t0

```

Another way do not use loop is like this,

```

    lw $t0, 0xupper
    add $t0, $t0, $t0
    add $t0, $t0, $t0
    add $t0, $t0, $t0
    add $t0, $t0, $t0
    add $t0, $t0, $t0
    add $t0, $t0, $t0
    add $t0, $t0, $t0
    add $t0, $t0, $t0
    add $t0, $t0, $t0
    add $t0, $t0, $t0
    add $t0, $t0, $t0
    add $t0, $t0, $t0
    add $t0, $t0, $t0
    add $t0, $t0, $t0
    add $t0, $t0, $t0
    add $t0, $t0, $t0
    add $t0, $t0, $t0
    add $t0, $t0, $t0
    lw $t1, 0xlower
    add $t1, $t0, $t1

```

- (E) lui requires an extra MUX and an extra control bit in the MIPS datapath, but it can load a large constant with only two instructions instead of long instructions in (D). Besides, with lui we can load arbitrary number of constants.

Problem 2. “Curse or Recurse” [35 points]

(A)

```

main: addi $a0, $0, 12    # load x to $a0
      addi $a1, $0, 24    # load y to $a1
      jal gcd
      beq $0, $0, end

gcd:
      addi $sp, $sp, -24   # ALLOCATE 6
      sw   $ra, -20($sp)
      bne  $a0, $a1, L1    # if (x == y)
      move $v0, $a0        # return x;
      beq  $0, $0, L4

L1:
      slt  $t0, $a0, $a1   # if (x < y)
      beq  $t0, $0, L2     # {
      sub  $a1, $a1, $a0   #   y = y - x;
      beq  $0, $0, L3     # }

```


Problem 3. “Be Fruitful and Multiply” [20 points]

```
mul:
    addiu $sp, $sp, -24
    #assume both $t0,$t1,$t2 are initialized to 0
    # $t0 used as a counter
    # $t1 as the result
    # $t2 as a flag

setup:
    bqt $a1, $0, loop
    sub $a1, $0, $a1    #set $a1 positive
    add $t2, $t2, 1    #set the negative flag

loop:
    beq $t0, $a1, endloop
    add $t0, $t0, 1    #update counter
    add $t1, $t1, $a0 #update sum
    j loop

endloop:
    beq $t2, $0, end
    sub $t1, $0, $t1

end:
    add $v0, $t0, $0
    addiu $sp, $sp, 24
    j $ra
```

Problem 4. “Benchmarking Bandwagon” [20 points]

(A) There are totally 1149 reads and 507 writes. So, percentage of all memory access

that are data access: $\frac{1149 + 507}{3751 + 1149 + 507} = 30.6\%$

Percentage of read in data access: $\frac{1149}{1149 + 507} = 69.4\%$

Percentage of all memory accesses that are reads: $\frac{3751 + 1149}{3751 + 1149 + 507} = 90.6\%$

(B) There are totally $(532+40+41+61)= 674$ instructions that change the PC. Hence,

the average basic block length is $\frac{3751}{674} = 5.56$

(C) Suppose that $x\%$ of the load operations are replaced by new instructions. Then,

$$(3751 - 1149 * x\%) * 1.1T_{cycle} = 3751 * T_{cycle}$$

The above equation gives us $x = 29.6$

(D) Suppose all 1149 origin loads are eliminated by all new instructions in (C), then performance speedup is at most $\frac{3751}{(3751-1149)*1.1} = 1.31$