CSEE W3827

Fundamentals of Computer Systems

Homework Assignment 3

Solutions

Prof. Stephen A. Edwards

Columbia University

Due June 30, 2015 at 5:30 PM

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. (20 pts.) In MIPS assembly, implement the `strncat` function from the C standard library, i.e.,

   ```
   char *strncat(char *dest, const char *src, size_t n)
   ```

   This appends at most *n* characters from the src string to the end of the dest string, overwriting dest's terminating 0 and returning dest. The returned string is always 0-terminated. Assume src and dest do not overlap. Assume dest, src, and n are each 32 bits.

   Start from the `strncat.s` template on the class website.

   Your function must obey MIPS calling conventions.

   Implement your function in the SPIM simulator.

   Turn in your solution on paper with evidence that it works. Add at least one test case.

   On the supplied test harness, your code should print

   ```
   Hello World!
   Hello World!

   Hello Wo
   3827 sucks you in with wonderful ideas
   3827 sucks you in with wonderful ideas

   3827 sucks you in
   ```

```
        # $a0 : dest
        # $a1 : src
        # $a2 : n
strncat:
        move    $v0, $a0     # return dest
        b       .L2          # Find the end of dest
.L1:    addiu   $a0, $a0, 1
.L2:    lbu     $t0, 0($a0)
        bnez    $t0, .L1
        b       .L4          # Start copying src to dest
.L3:    sb      $t0, 0($a0)  # Store character
        addiu   $a0, $a0, 1  # Next dest
        addiu   $a1, $a1, 1  # Next src
        addiu   $a2, $a2, -1 # n--
.L4:    beqz    $a2, .L5     # already copied n?
        lbu     $t0, 0($a1)  # Read source character
        bnez    $t0, .L3     # Hit the end of the string?
.L5:    li      $t0, 0       # Write terminator
        sb      $t0, 0($a0)
        jr $ra
```

2. (30 pts.) In MIPS assembly, implement a function that returns the maximum sum of node values from the root of a tree. Each node is represented by the consecutive words in memory: the value of the node (unsigned), the address of the left child, and the address of the right child. At each node, consider the maximum sum returned from the left and the right child and return it plus the node's value.

   Start from the `maxpath.s` template on the class website.

   Your function must obey MIPS calling conventions. Use the stack to implement the recursion.

   Implement your function in the SPIM simulator.

   Turn in your solution on paper with evidence that it produces the desired result. Add at least one test case.

   On the supplied test harness, your code should print

```
(42  )
42
(39 (3  ) (2  ))
42
(31 (42 (23  ) (31  )) (57 (1  ) ))
104
(6 (5 (3  ) (4 (1  ) (2  ))) (4 (1  ) (2  )))
17
```

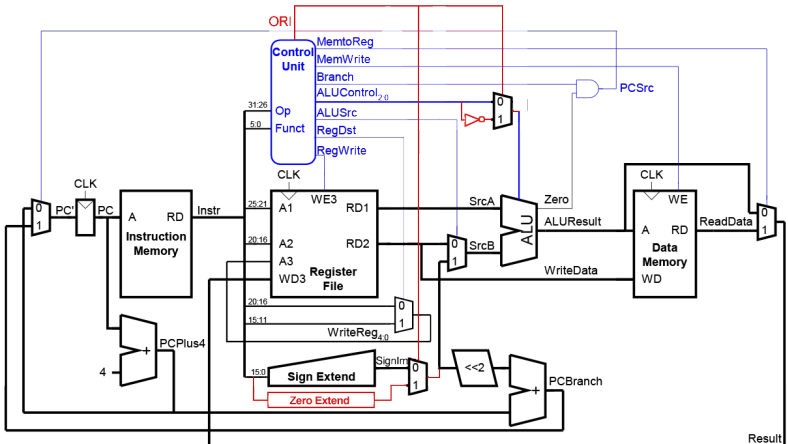
```
maxpath:
     bnez    $a0, .L1        # Return 0 at a null pointer
     li      $v0, 0
     jr      $ra
.L1: addi    $sp, $sp, -12
     sw      $ra, 0($sp)
     sw      $s0, 4($sp)
     sw      $s1, 8($sp)

     move    $s0, $a0        # Save the tree pointer
     lw      $a0, 4($s0)     # Recurse on left child
     jal     maxpath
     move    $s1, $v0        # Save left result in $s1
     lw      $a0, 8($s0)     # Recurse on right child
     jal     maxpath
     slt     $t0, $v0, $s1   # Pick the larger of the two
     beqz    $t0, .L2
     move    $v0, $s1

.L2: lw      $t0, 0($s0)
     addu    $v0, $v0, $t0   # add our node's value

     lw      $ra, 0($sp)
     lw      $s0, 4($sp)
     lw      $s1, 8($sp)
     addi    $sp, $sp, 12
     jr      $ra
```

3. (25 pts.) Extend the single-cycle MIPS processor to support the `ori` instruction (i-type, OP=001101).



Inst.	OP	RegWrite	RegDst	ALUSrc	Branch	MemWrite	MemToReg	ALUOp	ORI
R-type	000000	1	1	0	0	0	0	1-	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
ori	001101	1	0	1	0	0	0	01	1

4. (10 pts.) Assuming the following dynamic instruction frequency for a program running on the single-cycle MIPS processor

add	25%
addi	25%
beq	10%
lw	25%
sw	15%

(a) (5 pts.) In what fraction of all cycles is the data memory accessed (either read or written)?

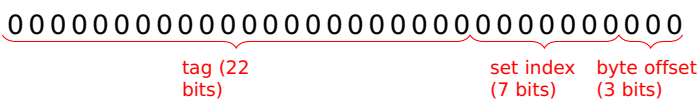
Only for loads and stores, so 25% (lw) + 15% (sw) = 40%.

(b) (5 pts.) In what fraction of cycles is the sign extend circuit used?

addi uses it for the immediate operand
beq uses it to compute the PC-relative address
lw uses it to compute the offset address
sw uses it to compute the offset address
So, 25% + 10% + 25% + 15% = 75%.

5. (15 pts.) For each of the caches listed below, show how a 32-bit addresses breaks into *tag*, *set index*, and *byte offset* fields.

Cache A: 4096B, 4-way set-associative, 8B lines
32B per set, so 128 sets in cache



Cache B: 2048B, direct-mapped, 16B lines
16B per set, so 128 sets in cache