CSEE W3827

Fundamentals of Computer Systems

Homework Assignment 3

Prof. Stephen A. Edwards

Columbia University

Due June 22, 2016 at 5:30 PM

Name:

Uni:

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 standard C function `rindex`:

   `char *rindex(const char *s, int c)`

   This returns a pointer to the *rightmost* occurrence of the character *c* in the string *s* or NULL if the character is not found. The terminating null byte is considered to be part of the string.

   Start from the `rindex.s` template on the class website; use the SPIM simulator.

   Your function must obey MIPS calling conventions.

   Turn in your solution on paper with evidence that it works. Add some test cases. Also, upload your solution as a single .s file to Courseworks.

   On the supplied test harness, your code should print

```
Looking for 'e' in "Hello World!"
Found at position 1
Looking for 'l' in "Hello World!"
Found at positio<n 9
Looking for 'z' in "Hello World!"
Not found
Looking for 'Hello World!"
Found at position 12
Looking for 'z' in "The quick brown fox jumps over the lazy dog"
Found at position 37
```

2. (30 pts.) In MIPS assembly, implement an "eval" function that walks a binary tree that represents an arithmetic expression and computes its meaning. Each tree node begins with a byte that indicates the the node is an integer (leaf) or operator plus two pointers to their arguments. In C, this would be

```
struct expr {
  char op; /* 0 for leaf */
  union {
      int leaf;
      struct {
        struct expr *left, *right;
      } branch;
  } pl;
};
```

```
int eval(struct expr *e)
{
  int left, right;
  if (e->op == 0) return e->pl.leaf;
  left = eval(e->pl.branch.left);
  right = eval(e->pl.branch.right);
  switch (e->op) {
  case '+': return left + right;
  case '-': return left - right;
  case '*': return left * right;
  }
  return 0;
}
```

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

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

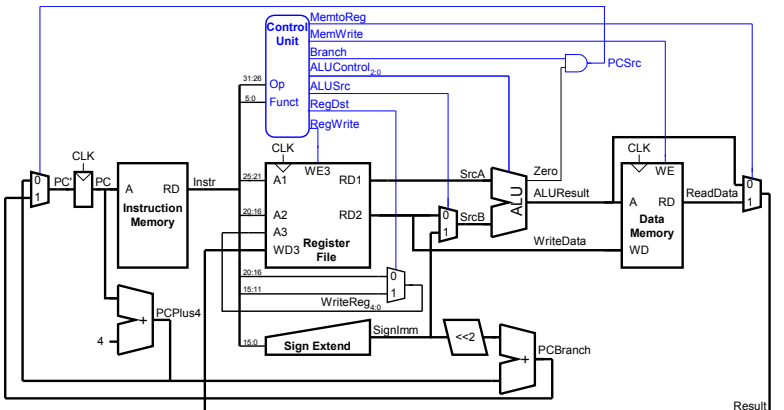
Implement your function in the SPIM simulator.

Turn in your solution on paper with evidence that it works. Add some test cases. Also, upload your solution as a single .s file to Courseworks.

On the supplied test harness, your code should print

```
42 = 42
17 = 17
25 = 25
(17+25) = 42
(5*(2+3)) = 25
((5*(2+3))+(42-17)) = 50
```

3. (25 pts.) Extend the single-cycle MIPS processor to support the `andi` instruction (i-type, OP=001100).



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

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

`addu`	25%
`addi`	25%
`beq`	15%
`lw`	20%
`sw`	15%

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

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

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: 8192B, 4-way set-associative, 16B lines

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Cache B: 4096B, direct-mapped, 8B lines

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0