1	
2	
3	
4	
5	

CSEE W3827

Fundamentals of Computer Systems

Homework Assignment 3

Prof. Stephen A. Edwards

Columbia University

Due June 26, 2017 at 1:00 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 `strspn`:

   `size_t strspn(const char *s, const char *accept)`

   This returns the length (in bytes) of the initial segment of the string *s* that consists entirely of bytes in *accept*.

   Start from the `strspn.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.

   On the supplied test harness, your code should print

```
strspan("Hello World!", "Hloe") = 5
strspan("Hello World!", "H Wdelor") = 11
strspan("Hello World!", "!H Wdelor") = 12
strspan("Hello World!", "HHHHHHllllooo") = 1
strspan("Hello World!", "HHHHHHoooeeeelll") = 5
strspan("Hello World!", "HHHHHHoooeeeelll Wrld!") = 12
strspan("", "Hello World!") = 0
strspan("Hello World!", "") = 0
strspan("Hello World!", "Hello World!") = 12
strspan("Hello World!", "HelWrld!") = 4
strspan("", "") = 0
```

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

```
struct expr {
  char op; /* 0 for leaf */
  union {
     unsigned int leaf;
     struct {
       struct expr *left;
       struct expr *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);
  if (e->op == '!') return ~left; /* bitwise NOT */
  right = eval(e->pl.branch.right);
  switch (e->op) {
  case '+': return left | right; /* bitwise OR */
  case '*': return left & right; /* bitwise AND */
  case '^': return left ^ right; /* bitwise XOR */
  }
  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.

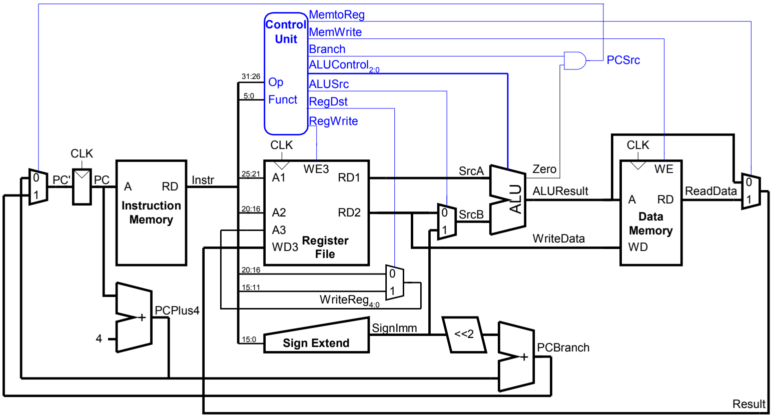
Implement your function in the SPIM simulator.

Turn in your solution on paper with evidence that it works. Add some test cases.

On the supplied test harness, your code should print

```
1 = 1
0 = 0
4 = 4
(0+4) = 4
(0+1) = 1
(1+1) = 1
(1+4) = 5
(7*(1+1)) = 1
(2+3) = 3
((1+4)+(0+1)) = 5
!(3) = -4
(0^0) = 0
(0^1) = 1
(1^1) = 0
((0+4)^(1+4)) = 1
!(((0+4)^(1+4))) = -2
```

3. (25 pts.) Extend the single-cycle MIPS processor to support the `xori` instruction (i-type, OP=001110).



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`	20%
`beq`	10%
`lw`	25%
`sw`	20%

(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: 16384B, 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: 8192B, direct-mapped, 32B 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