CS360 Lab #5 -- Assembler

- **Jim Plank**
- **CS360**

Stack/Register Templates

Suitable for printing in various formats:

- /blugreen/homes/plank/cs360/labs/lab5/template.ps -- postscript.
- /blugreen/homes/plank/cs360/labs/lab5/template.pdf -- pdf.
- /blugreen/homes/plank/cs360/labs/lab5/template.gif -- gif.

This is a written lab --- no programming. The TAs will tell you how best to get your answers to them. In these labs, you assume the architecture and instructions given in the lecture notes. Do *not* worry about delay slots. (In other words, I do not want to see any **noop** instructions).

Do not "optimize" the assembler. Give me the simple yet inefficient assembler that the compiler would return.

Question 1

Show the assembler that the following code compiles into:

```
int b(int j)
{
  int i;

  i = 10;
  while (i > 0) {
    j = j + i * i - 3;
    i--;
  }
  return j;
}
```

Question 2

Show the assembler that the following code compiles into. You may assume that the value of **NULL** is zero.

```
int f2(int *x)
{
```

```
  int j;

  if (x == NULL) return 48;

  j = *x * 2;
  return j;
}
```

Question 3

1. Show the assembler that the following code compiles into.

```
2. int b(int i, int j)
3. {
4.   if (j <= 0) return i+1;
5.
6.   return a(i, j-1)+1;
7. }
8.
9. int a(int i, int j)
10.  {
11.    if (i <= 0) return j+1;
12.    return b(i-1, j)+b(i-1, j-1);
13.  }
14.
15.  main(int argc, char **argv)
16.  {
17.    int i;
18.
19.    if (argc != 3) { exit(1); }
20.    i = a(atoi(argv[1]), atoi(argv[2]));
21.  }
```

22. What is the final value of **i** when this program is called in the following ways:

```
23.  UNIX> a.out 0 0
24.  UNIX> a.out 1 2
25.  UNIX> a.out 2 1
```

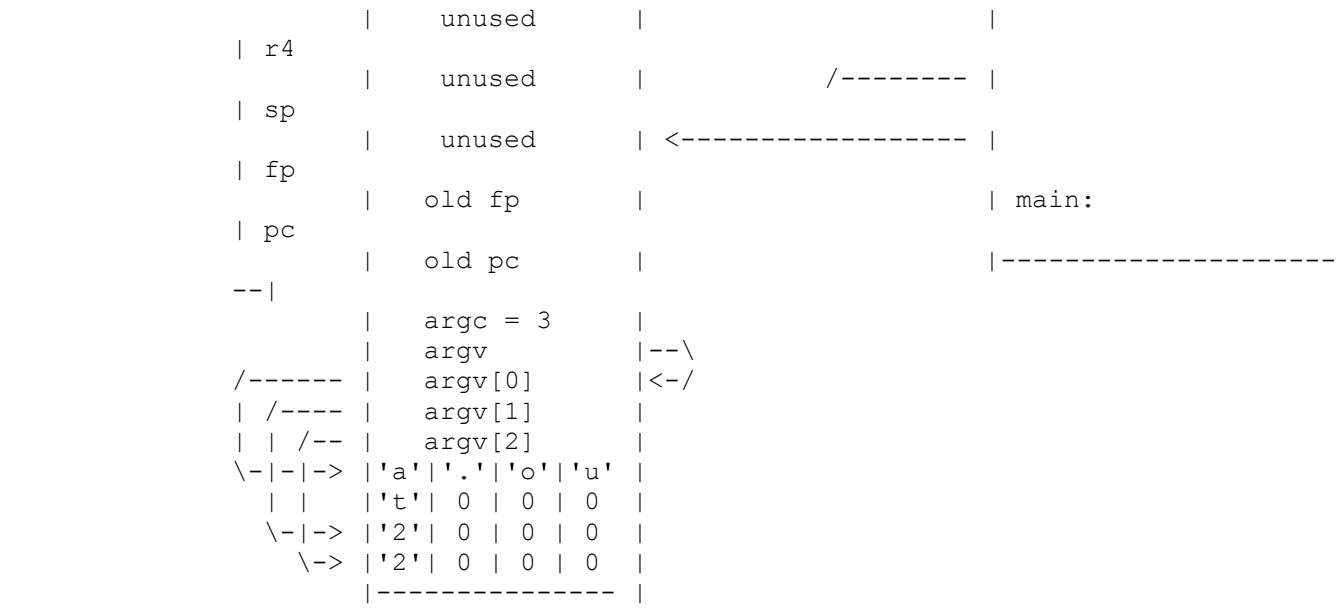
26. Show the state of the stack and registers the first six times a **jsr** statement is reached when the program is called as follows:

```
27.  UNIX> a.out 2 2
```

Assume that the state of the stack and registers when main is first called looks as follows.

Note, I'm happy with arrows instead of numeric values.

```
                 Stack                             registers
          |----------------|                 |---------------------
--|
          |                |                 |
| r0
          |                |                 |
| r1
          |     .....      |                 |
| r2
          |    unused      |                 |
| r3
```



Question 4

1. Show the assembler that the following code compiles into (assume that the compiler doesn't check that the call of **a** in **b** has the wrong number of arguments).

```
int a(int i, int j, int *k)
{
  if (i == 0) {
    *k = 15;
    return j
  } else {
    return j;
  }
}

int b(int i)
{
  int m;

  m = i+25;

  return a(i);
}

main(int argc, char **argv)
{
  int i;
  i = b(atoi(argv[1]));
}
```

2. Show the state of the stack right before **a** returns for each of the following invocations of the program:

```
UNIX> a.out 0
UNIX> a.out 1
```

3. In each of the invocations above, what is the final value of **i**?

Fri Feb 22 10:27:04 EST 2002