CS 360 LAB Assignment #1

DUE: Friday, 1-29-2016

OBJECTIVES: Stack Usage and YOUR myprintf() FUNCTION

An operating system supports only how to print char. All other printings are based on printing char. In Linux, putchar(char c); prints a char.

1. Use putchar() to write YOUR OWN prints(char *s) to print a string.

2. typedef unsigned int u32;

   The following printu(u32 x) function prints an unsigned int (u32)x:

```
int BASE = 10;

char *table = "0123456789ABCDEF";

int rpu(u32 x)
{
  char c;
  if (x){
     c = table[x % BASE];
     rpu(x / BASE);
     putchar(c);
  }
}

int printu(u32 x)
{
  if (x==0)
     putchar('0');
  else
     rpu(x);
  putchar(' ');
}
```

3. WRITE YOUR OWN functions:
   int printd(int x): print an integer (which may be negative!!)
   int printo(u32 x): print x in OCTal as 0.....
   int printx(u32 x): print x in HEX. as 0x....

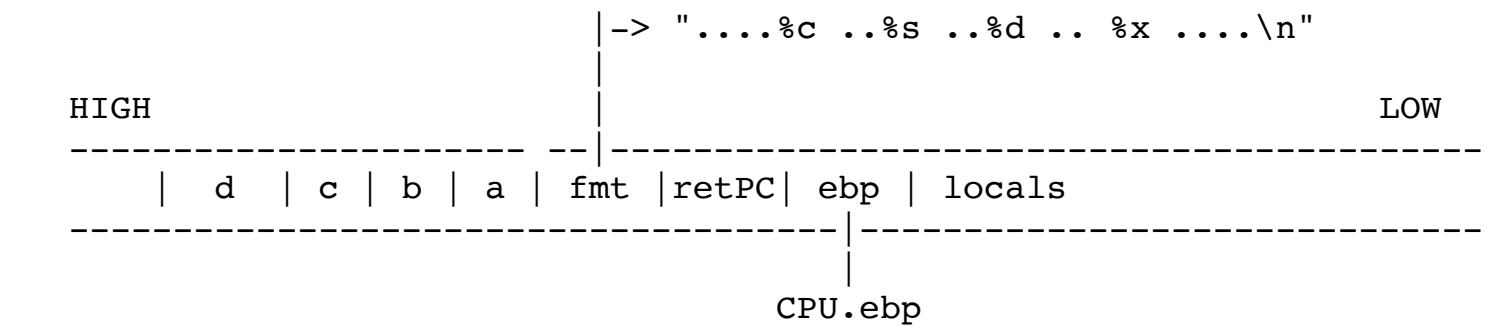
4. WRITE a myprintf(char *fmt, ...) for formatted printing, e.g.

```
      myprintf("this is a string\n");    // PRINT: this is a string
      myprintf("this is %c hi %d test %x\n", 'A', 1234, 100);

      // PRINT: this is A hi 1234 test 0x64
```

  Your myprintf() should print items by %c, %s, %u, %d, %o, %x,
  just like printf().

```
/********************* HELP INFO ***********************************
int myprintf(char *fmt, ...) // SOME C compiler requires the 3 DOTs
{
   Assume the call is myprintf(fmt, a,b,c,d);
   Upon entry, the stack contains:
```



1. Let char *cp point at the format string: char *cp = fmt;

2. Let int *ip point at the first item to be printed on stack:
   int *ip = &fmt + 1;

********** ALGORITHM **********
Use cp to scan the format string:
    spit out any char that's NOT %
    for each \n, spit out an extra \r

Upon seeing a %: get next char, which must be one of 'c','s','u','d', 'o','x'
Then call

```
    putchar(*ip) for 'c';
    prints(*ip)  for 's';
    printu(*ip)  for 'u';
    printd(*ip)  for 'd';
    printo(*ip)  for 'o';
    printx(*ip)  for 'x';
```

  Advance ip to point to the next item on stack.

After implementing your myprintf() function, write simple C programs to test your myprintf() function first.

5. Given: s.s file:

```
#--------------------- s.s file --------------------------
        .global main, mymain, myprintf
main:
        pushl  %ebp
        movl   %esp, %ebp

# (1). Write ASSEMBLY code to call myprintf(FMT)
#      HELP: How does mysum() call printf() in the class notes.


# (2). Write ASSEMBLY code to call mymain(argc, argv, env)
#      HELP: When crt0.o calls main(int argc, char *argv[], char *env[]),
#            it passes argc, argv, env to main().
#            Draw a diagram to see where are argc, argv, env?


# (3). Write code to call myprintf(fmt,a,b)
#      HELP: same as in (1) above


# (4). Return to caller
       movl  %ebp, %esp
```

```
        popl  %ebp
        ret

#---------- DATA section of assembly code ---------------
        .data
FMT:    .asciz "main() in assembly call mymain() in C\n"
a:      .long 1234
b:      .long 5678
fmt:    .asciz "a=%d b=%d\n"
#---------  end of s.s file ---------------------------


/********************* t.c file *********************************

mymain(int argc, char *argv[ ], char *env[ ])
{
  int i;

  myprintf("in mymain(): argc=%d\n", argc);

  for (i=0; i < argc; i++)
      myprintf("argv[%d] = %s\n", i, argv[i]);

  // WRITE CODE TO PRINT THE env strings

  myprintf("---------- testing YOUR myprintf() ---------\n");
  myprintf("this is a test\n");
  myprintf("testing a=%d b=%x c=%c s=%s\n", 123, 123, 'a', "testing");
  myprintf("string=%s, a=%d  b=%u  c=%o  d=%x\n",
           "testing string", -1024, 1024, 1024, 1024);
  myprintf("mymain() return to main() in assembly\n");
}


6. Run    gcc -m32 t.c s.s             to generate a.out
   Run
          a.out one two three four

   to test your main, mymain() and myprintf()
============================================================

7. Sample Solutions: samples/LAB1/
                              |
                            lab1.bin
                            lab1.static
```