myprintf.c

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>

void printx(int);
void printd(int);

int myprintf(char* format, ...) {

 int len = strlen(format);
 int *arg;
 arg = (int*)&format;
 int i;
 int j;
 arg++;

 for(i = 0; i < len; i++) {
  if(format[i] == '%') {
     if(format[i+1] == 'c') {
          int b = *arg;
          arg++;
          putchar(b);
          i++;
     }
     else if(format[i+1] == 's') {
       char *string;
          string = *((char**)arg);
          int length = strlen(string);
       //printf("%d\n", length);
       for(j = 0; j < length; j++) {
            putchar(string[j]);
          }
          i++;
          arg++;
     }
     else if(format[i+1] == 'x') {
          int x = *arg;
       printx(x);
       arg++;
       i++;
     }
     else if(format[i+1] == 'd') {
          int d = *arg;
       printd(d);
       arg++;
       i++;
     }
     else if(format[i+1] == '%') {
        putchar('%');
```

```
    }
  }
  else {
    putchar(format[i]);
  }
 }
}
```

printinteger.s

```
.section .data
.section .text

u_divide_by_10:

/* r0 contains the argument to be divided by 10 */

   ldr r1, .Lu_magic_number_10      /* r1 ← magic_number */
   umull r1, r2, r1, r0             /* r1 ← Lower32Bits(r1*r0). r2 ← Upper32Bits(r1*r0) */
   mov r0, r2, LSR #3               /* r0 ← r2 >> 3 */
   b next                           /* leave function */
   .align 4
   .Lu_magic_number_10: .word 0xcccccccd


.global printx
.global printd

printx:
        push {fp,lr}
      mov r3, r0
      mov r5, #0
lab1:
      lsl r0, r0, #28
      lsr r0, r0, #28
      mov r2, #10
      cmp r0, r2
      blt lab2
      add r0, r0, #87
      b lab3

lab2:
      add r0, r0, #48

lab3:
      push {r0}
      add r5, r5, #1
      lsr r3, r3, #4
      mov r4, #0
      mov r0, r3
```

```
        cmp r3, r4
        bne lab1
        b lab4
lab4:
        pop {r0}
        sub r5, r5, #1
        bl putchar
        cmp r5, #0
        bne lab4

        pop {fp,pc}


printd:
            push {fp,lr}
start:
        mov r3, r0
        lsr r3, r3, #31
            cmp r3, #1
            bne start2
        mov r10, #0xFFFFFFFF
        sub r0, r10,r0
        add r0, r0, #1
        mov r10, #100
        b start2

start2:
        mov r3, r0
        mov r4, r0
        mov r5, #0
        mov r9, #10

lab7:
            b u_divide_by_10
next:
            mov r3, r0
        mul r8, r0, r9
        sub r7, r4, r8
        push {r7}
        add r5, r5, #1
        cmp r3, #0
            mov r4, r3
        mov r0, r3
        bne lab7
        cmp r10, #100
        bne lab8
        mov r10, #-3
        push {r10}
        add r5, r5, #1
lab8:
        pop {r0}
```

```
    sub r5, r5, #1
    add r0, r0, #48
    bl putchar
    cmp r5, #0
    bne lab8

    pop {fp,pc}
```