EECS 373 Fall 2017 Homework #3

Due September 25th on Gradescope. Late homework is not accepted.

Name: _____ unique name: _____

You are to turn in this assignment filling in the blanks as needed. ***Assignments that are not done on this worksheet or are difficult to read will lose at least 50% of the possible points and we may not grade them at all.*** This is an individual assignment; all work should be your own. 50 points.

---

1) Loaders, Linkers, and Executables
   a) In five or fewer sentences, explain the role of the assembler and linker.

b) Consider the Thumb below. Briefly explain why the label “eoa” is at a higher address than the label “stop” even though the label “eoa” appears first. Note, you do not have to fill up all the space. (Source code modified from http://www.bravegnu.org/gnu-eprog/lds.html).

```
1         .data                   @ Address: 0x00000400
2  arr:   .byte 10, 20, 25        @ Read-only array of bytes
3  eoa:                           @ Address of end of array + 1
4
5         .text                   @ Address: 0x00000000
6  start:
7         ldr   r0, eoa           @ r0 = &eoa
8         ldr   r1, arr           @ r1 = &arr
9         mov   r3, #0            @ r3 = 0
10 loop:  ldrb  r2, [r1], #1      @ r2 = *r1++
11        add   r3, r2, r3        @ r3 += r2
12        cmp   r1, r0            @ if (r1 != r2)
13        bne   loop              @    goto loop
14 stop:  b stop
```

2) Write rewrite the function “A” in UAL assembly which does the same thing as the following C code. You should assume “print” is some ABI compliant function which takes a single integer argument and does *something* with “a.” “B” is also an ABI compliant function. (Note, one point of the ABI is to be able to mix C and assembly like this, the linker will make it all work!)

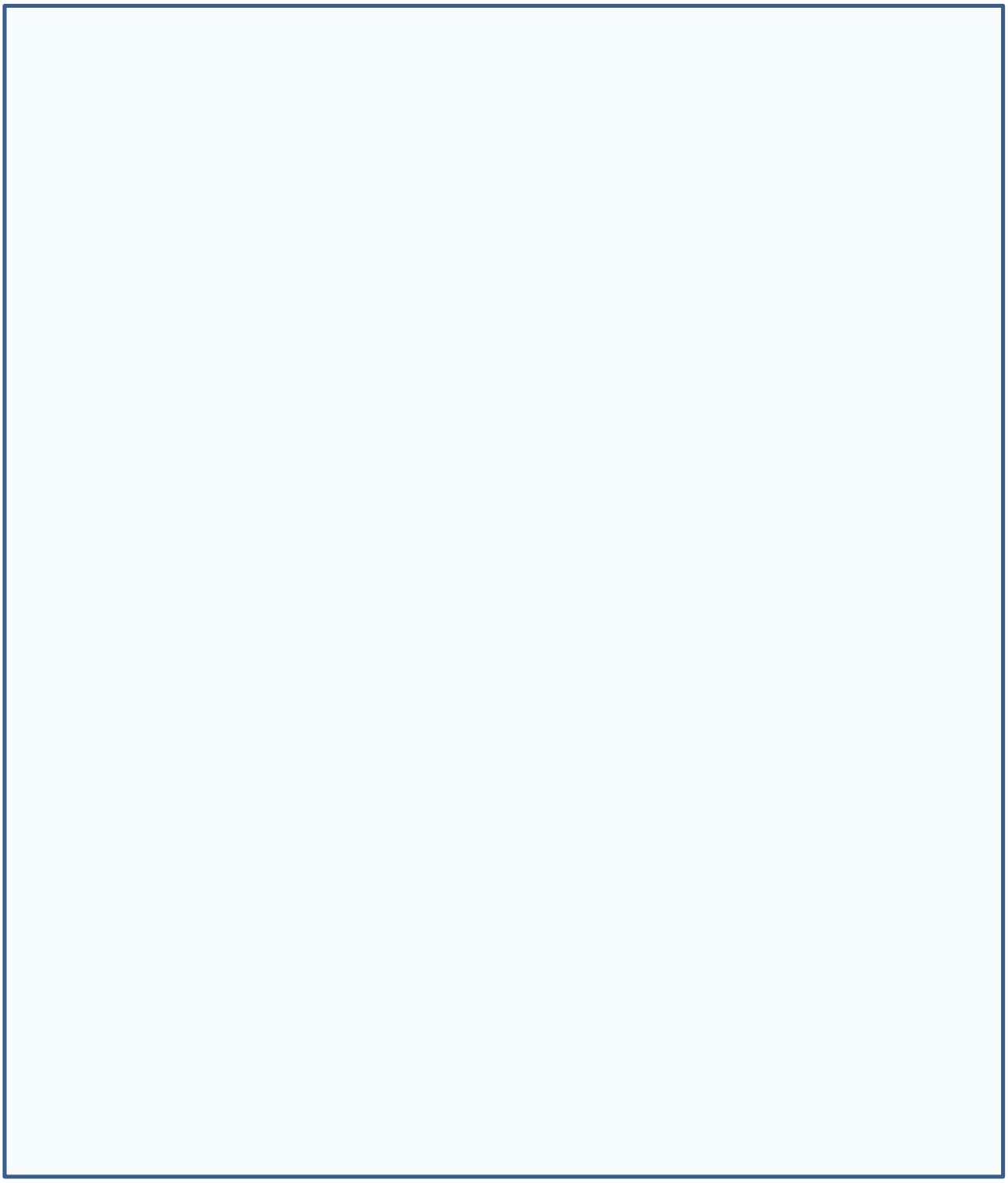
```
void main(void)
{
  int a=2,b=6;

  b=A(a,b);
  a=a*b;
  print(b);
  print(a);

}
int A(int x, int y)
{
  int a;
  a=B(x+y, x-y);
  print(a);
  a=a*(x+y)
  return(a+x-y);
}
```

3) Write a short UAL assembly program which every time the button is pressed toggles the light (so that if it's off pressing the button turns it back on and if it's on pressing the button turns it off. You can assume the switch is debounced. Be sure that pressing the button doesn't cause the light to quickly flicker—rather each button press (off to on) should cause the light to toggle. The button is at address 0x5 and the LED is at address 0x4. Assume the button and LED are active high and that code for initialization is already written for you.

```
@assume code for initialization provided
main:
```

a) The same as part a, but in C this time.

4) Draw a picture of two APB slave devices and an APB master device. Draw each of the three devices with all inputs on the left and all outputs on the right. It should be clear where wires are shared by the slave devices and where they have their own wires. Include all the wires/busses found in the APB specification, which can be found on the course website. You may attach an image or print, complete, and scan this page.



5) Assume you have a memory-mapped register location REG_FOO on a 32-bit architecture. Write the additional C code needed to toggle bit 22 of the word at that location (that is change it from a 1 to a 0 or a 1 to a 1). The code should be written so that no complier errors occur. **[10 points]**

REG_FOO location: 0x4005000