CISC 260

Solution set: HW6

Exercise 1

Executable file header		
	Text size	0x440
	Data size	0x90
Text segment	Address	Instruction
	0x0000 8000	LDR r0, [r3, #0]
	0x0000 8004	ORR r1, r0, #0
	0x0000 8008	BL 0x0000 004C
	-	-
	0x0000 8140	STR r0, [r3, #0]
	0x0000 8144	B 0x0000 005D
	-	-
	0x0000 8320	MOV pc, (FFFF FFA1)
	-	-
Data segment	Address	
	0x1000 0000	X
	0x0000 8140	B
	0x1000 0040	Y
	0x0000 82C0	FOO

Exercise 2

a. $Initial\ execution\ time = (500 + 300 \times 10 + 100 \times 3) \times 1 = \mathbf{3800}$
$New\ execution\ time = (500 \times 0.75 + 300 \times 10 + 100 \times 3) \times 1.1 = \mathbf{4042.5}$
Not a good design.

b. By doubling the performance of arithmetic instructions, the Clock cycle of arithmetic instructions are reduced to 0.5.
Therefore, $CPU\ time = (500 \times 0.5 + 300 \times 10 + 100 \times 3) \times 1 = 3550$

$$Speedup = \frac{3800}{3550} = \mathbf{1.0704}$$

By improving the performance of arithmetic instructions by 10 times, the Clock cycle of arithmetic instructions are reduced to 0.1 (by a factor of 10).
$CPU\ time = (500 \times 0.1 + 300 \times 10 + 100 \times 3) \times 1 = 3350$

$$Speedup = \frac{3800}{3350} = \mathbf{1.134}$$

Exercise 3

a. $Average\ CPI = 2 \times 0.7 + 6 \times 0.1 + 3 \times 0.2 = \mathbf{2.6}$

b. On 25% improvement, average CPI becomes, $\frac{2.6}{1.25} = 2.08$

$$Arithmetic\ CPI = \frac{2.08 - (6 \times 0.1 + 3 \times 0.2)}{0.7} = \mathbf{1.257}$$

c. On 50% improvement, average CPI becomes, $\frac{2.6}{1.5} = 1.73$

$$Arithmetic\ CPI = \frac{1.73 - (6 \times 0.1 + 3 \times 0.2)}{0.7} = \mathbf{0.762}$$

Exercise 4

The idea is to first convert the C function so that the recursion is tail recursion.

```
funct( int x, int total, int count ){
      if( count > x ) return total;
      else if( count == x && x & 0x01 ){
         return total + x;
      }
      else if( count & 0x01 ){
         return funct( x, count + total, count + 1 );
      }
      else {
         return funct( x, count - total, count + 1 );
      }
}
```

The rest part is simply converting it into ARM assembly code. There could be different implementations.