Lab 2 Prelab

David Chun (dc37875) and Grace Zhuang (gpz68)

2. The microcontroller is executing at 80 MHz. Assuming assembly instructions average about 2 cycles per instruction, it takes about 25 ns to execute an instruction. Look at the following C code and associated assembly created by the compiler. Answer the following questions

a) What is the purpose of all the DCW statements?

They fill the memory location with an initial value.

b) The main program toggles PF1. Neglecting interrupts for this part, estimate how fast PF1 will toggle.

PF1 should toggle once every 150ns (25ns * 6 assembly instructions).

c) What is in R0 after the first LDR is executed? What is in R0 after the second LDR is executed?

After the first LDR, R0 = 0x40025000. After the second LDR, R0 = 0x00000000.

d) How would you have written the compiler to remove an instruction?

In the first instruction, instead of LDR R0, [PC, #24], we could LDR into R1 instead. This way, the instruction at 0x00000686 can be taken out. R1 will hold the address of PF1, and R0 will hold the value of PF1.

e) 100-Hz ADC sampling occurs in the Timer0 ISR. The ISR toggles PF2 three times. Toggling three times in the ISR allows you to measure both the time to execute the ISR and the time between interrupts. See Figure 2.1. Do these two read-modify write sequences to Port F create a critical section? If yes, describe how to remove the critical section? If no, justify your answer?

No, these two read-modify write sequences to Port F do not create a critical section. Both write sequences use bit-specific addressing. The ISR toggles PF2, while the main program toggles PF1. At any given time, only one of the bits will be written to.

This assembly code was obtained by observing the assembly listing in the debugger. You may see different assembly on your machine because of differences in the compiler version or optimization settings. You are allowed to solve the preparation with either this assembly or the assembly you see on your computer.

```
0x0000067E 4806     LDR r0,[pc,#24]  ; @0x0000068C
0x00000680 6880     LDR r0,[r0,#0x08]
0x00000682 F0800002  EOR r0,r0,#0x02
0x00000686 4904     LDR r1,[pc,#16]  ; @0x0000068C
0x00000688 6088      STR r0,[r1,#0x08]
0x0000068A E7F8     B   0x0000067E
0x0000068C E608     DCW 0xE608
0x0000068E 400F     DCW 0x400F
0x00000684 5400     DCW 0x5400
0x00000686 4002      DCW 0x4002
0x00000688 551C     DCW 0x551C
0x0000068A 4002     DCW 0x4002
0x0000068C 5000     DCW 0x5000
0x0000068E 4002     DCW 0x4002
```

```
while(1){
     PF1 ^= 0x02;
}
```