Ava Zahedi
ME 333
HW for Class 12

Chapter 5 (use no optimization for all exercises): Exercises 3, 4.
Chapter 6: Exercises 1, 4, 5, 8, 9, 16,17. Make a demo video for 6.17

Chapter 5
3.

a. - Long long integer division,
   - Float addition, subtraction, multiplication, and division
   - Long double addition, subtraction, multiplication, and division

   For j3 = j1/j2;

   ```
   9d0086cc:   0f401e52    jal   9d007948 <__divdi3>
   Jumps to
   9d007948 <__divdi3>:
   9d007948:   04a10007    bgez   a1,9d007968 <__divdi3+0x20>
   ```

b. Integer addition, subtraction, and multiplication only have 4 assembly commands, the fewest out of all the possible combinations.

   Example:

   ```
   i3 = i1+i2;
   9d0085d0:   8fc30014    lw   v1,20(s8)
   9d0085d4:   8fc20018    lw   v0,24(s8)
   9d0085d8:   00621021    addu   v0,v1,v0
   9d0085dc:   afc2004c    sw   v0,76(s8)
   ```

   The smallest data type, char, is not involved. This is because the extra assembly command andi ANDs the char with 0xFF. This essentially converts it to a 32-bit value from an 8-bit value.

c.

	char	int	long long	float	long double
+	1.25 (5)	1.0 (4)	2.75 (11)	1.25 (5)	2.0 (8)
-	1.25 (5)	1.0 (4)	2.75 (11)	1.25 (5)	2.0 (8)
*	1.25 (5)	1.0 (4)	4.5 (18)	1.25 (5)	2.0 (8)
/	1.75 (7)	1.75 (7)	2.0 (8)	1.25 (5)	2.0 (8)

d.

Section	Address	Length (bytes)	Length (bytes decimal)
.text	0x9d007948	0x444	1092
.text.dp32subadd	0x9d007d8c	0x430	1072
.text.dp32mul	0x9d0081bc	0x32c	812
.text	0x9d0084e8	0x5ac	1452
.text.fpsubadd	0x9d008a94	0x278	632
.text.fp32div	0x9d008d0c	0x230	560
.text.fp32mul	0x9d008f3c	0x1bc	444

4.

- & uses 4 commands
- | uses 4 commands
- << uses 3 commands
- >> uses 3 commands

Chapter 6

1. A benefit of interrupts is that the system doesn't have to continuously check while polling does. This is more efficient. A con of interrupts is that if you have multiple sources of interrupts, this can cause latency issues. A benefit of polling, however, is that the timing of checking for an event is much more predictable instead of having to wait on an external notification. However, this also means that it is continuously checking for an event, even when one isn't occurring.

4.

a. The CPU will execute the ISR immediately.
b. The CPU will pause the priority level 2, subpriority level 3 ISR and execute the priority level 4, subpriority level 2 ISR first.
c. The CPU will pause the priority level 4, subpriority level 0 ISR and execute the priority level 4, subpriority level 2 ISR first.
d. The CPU will continue executing the priority level 6, subpriority level 0 ISR until it is complete, then execute the incoming ISR after.

5.

a. The first thing the CPU must do before executing the ISR is copy register data to the data RAM (context save). The last thing it must do upon completing the ISR is restore the previous CPU state so that it can continue where it left off before the interrupt (context restore).
b. The shadow register set (SRS) is a full extra set of registers which eliminates the time needed for context save and restore. Instead of copying back and forth, the CPU can just switch to this extra set of registers for processing the ISR.

8.

a. IEC0SET = 0b100000000; // enable the Timer2 interrupt
   IFS0CLR = 0b100000000; // set the flag status to 0
   IPC2SET = 0b10110; // set the Timer2 interrupt priority to 5 and subpriority to 2

b. IEC1SET = 0b1000000000000000; // enable the RTCC interrupt
   IFS1CLR = 0b1000000000000000; // set the flag status to 0
   IPC8SET = 0b 11001000000000000000000000000000;
   // set the RTCC interrupt priority to 6 and subpriority to 1

c. IEC2SET = 0b10000; // enable the UART4 receiver interrupt
   IFS2CLR = 0b10000; // set the flag status to 0
   IPC12SET = 0b1111100000000;
   // set the UART4 receiver interrupt priority to 7 and subpriority to 3

d.
   IEC0SET = 0b100000000000; // enable the INT2 interrupt
   IFS0CLR = 0b100000000000; // set the flag status to 0
   IPC12SET = 0b01110000000000000000000000000000;
   // set the INT2 external input interrupt priority to 3 and subpriority to 2
   INTCONSET = 0b10 // configure to trigger on a rising edge

9. INT_timing_mod.c attached

16. INT_ext_int_mod.c attached

17. chp6_q17.c attached