Exercise 5.3

(a)

j6 = j1/j2; (divide of long long int)

```
9d003028:    8fc60028    lw     a2,40(s8)
9d00302c:    8fc7002c    lw     a3,44(s8)
9d003030:    8fc40020    lw     a0,32(s8)
9d003034:    8fc50024    lw     a1,36(s8)
9d003038:    0f4008ae    jal    9d0022b8 <__divdi3>
9d00303c:    00000000    nop
9d003040:    afc20078    sw     v0,120(s8)
9d003044:    afc3007c    sw     v1,124(s8)
```

All calculation of float

f3 = f1+f2;

```
9d003048:    8fc50034    lw     a1,52(s8)
9d00304c:    8fc40030    lw     a0,48(s8)
9d003050:    0f400d01    jal    9d003404 <__addsf3>
9d003054:    00000000    nop
9d003058:    afc20080    sw     v0,128(s8)
```

f4 = f1-f2;

```
9d00305c:    8fc50034    lw     a1,52(s8)
9d003060:    8fc40030    lw     a0,48(s8)
9d003064:    0f400cff    jal    9d0033fc <__subsf3>
9d003068:    00000000    nop
9d00306c:    afc20084    sw     v0,132(s8)
```

f5 = f1*f2;

```
9d003070:    8fc50034    lw     a1,52(s8)
9d003074:    8fc40030    lw     a0,48(s8)
9d003078:    0f400e29    jal    9d0038a4 <__mulsf3>
9d00307c:    00000000    nop
```

```
9d003080:	afc20088	sw	v0,136(s8)
  f6 = f1/f2;
9d003084:	8fc50034	lw	a1,52(s8)
9d003088:	8fc40030	lw	a0,48(s8)
9d00308c:	0f400d9d	jal	9d003674 <__divsf3>
9d003090:	00000000	nop
9d003094:	afc2008c	sw	v0,140(s8)
```

All calculation of long double

```
d3 = d1+d2;
9d003098:	8fc60040	lw	a2,64(s8)
9d00309c:	8fc70044	lw	a3,68(s8)
9d0030a0:	8fc40038	lw	a0,56(s8)
9d0030a4:	8fc5003c	lw	a1,60(s8)
9d0030a8:	0f4009c0	jal	9d002700 <__adddf3>
9d0030ac:	00000000	nop
9d0030b0:	afc20090	sw	v0,144(s8)
9d0030b4:	afc30094	sw	v1,148(s8)
  d4 = d1-d2;
9d0030b8:	8fc60040	lw	a2,64(s8)
9d0030bc:	8fc70044	lw	a3,68(s8)
9d0030c0:	8fc40038	lw	a0,56(s8)
9d0030c4:	8fc5003c	lw	a1,60(s8)
9d0030c8:	0f4009be	jal	9d0026f8 <__subdf3>
9d0030cc:	00000000	nop
9d0030d0:	afc20098	sw	v0,152(s8)
9d0030d4:	afc3009c	sw	v1,156(s8)
  d5 = d1*d2;
9d0030d8:	8fc60040	lw	a2,64(s8)
9d0030dc:	8fc70044	lw	a3,68(s8)
```

```
9d0030e0:     8fc40038     lw     a0,56(s8)
9d0030e4:     8fc5003c     lw     a1,60(s8)
9d0030e8:     0f400aca     jal    9d002b28 <__muldf3>
9d0030ec:     00000000     nop
9d0030f0:     afc200a0     sw     v0,160(s8)
9d0030f4:     afc300a4     sw     v1,164(s8)
  d6 = d1/d2;
9d0030f8:     8fc60040     lw     a2,64(s8)
9d0030fc:     8fc70044     lw     a3,68(s8)
9d003100:     8fc40038     lw     a0,56(s8)
9d003104:     8fc5003c     lw     a1,60(s8)
9d003108:     0f400780     jal    9d001e00 <__divdf3>
9d00310c:     00000000     nop
9d003110:     afc200a8     sw     v0,168(s8)
9d003114:     afc300ac     sw     v1,172(s8)
```

(b)

```
i3 = i1+i2;
9d002f3c:     8fc30014     lw     v1,20(s8)
9d002f40:     8fc20018     lw     v0,24(s8)
9d002f44:     00621021     addu   v0,v1,v0
9d002f48:     afc2004c     sw     v0,76(s8)
  i4 = i1-i2;
9d002f4c:     8fc30014     lw     v1,20(s8)
9d002f50:     8fc20018     lw     v0,24(s8)
9d002f54:     00621023     subu   v0,v1,v0
9d002f58:     afc20050     sw     v0,80(s8)
  i5 = i1*i2;
9d002f5c:     8fc30014     lw     v1,20(s8)
9d002f60:     8fc20018     lw     v0,24(s8)
```

```
9d002f64:    70621002    mul    v0,v1,v0
9d002f68:    afc20054    sw     v0,84(s8)
```

Char does not involve in the smallest data type. It has an additional line shown below.

```
9d002f18:    304200ff    andi   v0,v0,0xff
```

andi transfer the data types back to char by comparing the calculated result with 0xff through & operator

(c)

| | char | int | Long long | float | Long double |
|---|---|---|---|---|---|
| + | 1.25(5) | 1(4) | 2.75(11) | 2.25(5+4) J | 3(8+4) J |
| - | 1.25(5) | 1(4) | 2.75(11) | 1.5(5+1) J | 2.25(8+1) J |
| * | 1.25(5) | 1(4) | 4.5(18) | 2.25(5+4) J | 3.5(8+6) J |
| / | 1.75(7) | 1.75(7) | 70(8+272) J | 2.25(5+4) J | 5.25(15+6) J |

(d)

j6 = j1/j2;

```
.text          0x000000009d0022b8     0x440
 .text         0x000000009d0022b8     0x440 c:/program files/microchip/xc32/v4.00/bin/bin/../../lib/gcc/pic32mx/8.3.1\libgcc.a(_divdi3.o)
               0x000000009d0022b8              __divdi3
```

f3 = f1+f2;  f4 = f1-f2;

```
.text.fpsubadd  0x000000009d0033fc      0x278
 .text.fpsubadd
               0x000000009d0033fc      0x278 c:/program files/microchip/xc32/v4.00/bin/bin/../../lib/gcc/pic32mx/8.3.1/../../../../pic32mx/lib\libm.a(fp32subadd.o)
               0x000000009d0033fc              __subsf3
               0x000000009d0033fc              fpsub
               0x000000009d003404              __addsf3
               0x000000009d003404              fpadd
```

f5 = f1*f2;

```
.text.fp32mul   0x000000009d0038a4      0x1bc
```

```
 .text.fp32mul  0x000000009d0038a4      0x1bc c:/program files/microchip/xc32/v4.00/bin/bin/../../lib/gcc/pic32mx/8.3.1/../../../../pic32mx/lib\libm.a(fp32mul.o)
                0x000000009d0038a4                __mulsf3
                0x000000009d0038a4                fpmul
```

f6 = f1/f2;

```
.text.fp32div   0x000000009d003674      0x230
 .text.fp32div  0x000000009d003674      0x230 c:/program files/microchip/xc32/v4.00/bin/bin/../../lib/gcc/pic32mx/8.3.1/../../../../pic32mx/lib\libm.a(fp32div.o)
                0x000000009d003674                fpdiv
                0x000000009d003674                __divsf3
```

d3 = d1+d2;  d4 = d1-d2;

```
.text.dp32subadd
                0x000000009d0026f8      0x430
 .text.dp32subadd
                0x000000009d0026f8      0x430 c:/program files/microchip/xc32/v4.00/bin/bin/../../lib/gcc/pic32mx/8.3.1/../../../../pic32mx/lib\libm.a(dp32subadd.o)
                0x000000009d0026f8                dpsub
                0x000000009d0026f8                __subdf3
                0x000000009d002700                __adddf3
                0x000000009d002700                dpadd
```

d5 = d1*d2;

```
.text.dp32mul   0x000000009d002b28      0x32c
 .text.dp32mul  0x000000009d002b28      0x32c c:/program files/microchip/xc32/v4.00/bin/bin/../../lib/gcc/pic32mx/8.3.1/../../../../pic32mx/lib\libm.a(dp32mul.o)
                0x000000009d002b28                __muldf3
                0x000000009d002b28                dpmul
```

d6 = d1/d2;

```
.text.dp32mul   0x000000009d001e00      0x4b8
```

```
 .text.dp32mul  0x000000009d001e00      0x4b8 c:/program files/microchip/xc32/v4.00/bin/bin/../../lib/gcc/pic32mx/8.3.1/../../../../pic32mx/lib\libm.a(dp32div.o)
                0x000000009d001e00                dpdiv
                0x000000009d001e00                __divdf3
```

kseg0 Program-Memory Usage for math subroutine

| section | address | length [bytes] | (dec) | Description |
|---|---|---|---|---|
| .text.dp32mul | 0x9d001e00 | 0x4b8 | 1208 | |
| .text | 0x9d0022b8 | 0x440 | 1088 | App's exec code |
| .text.dp32subadd | 0x9d0026f8 | 0x430 | 1072 | |
| .text.dp32mul | 0x9d002b28 | 0x32c | 812 | |
| .text | 0x9d002e54 | 0x5a8 | 1448 | App's exec code |
| .text.fpsubadd | 0x9d0033fc | 0x278 | 632 | |
| .text.fp32div | 0x9d003674 | 0x230 | 560 | |
| .text.fp32mul | 0x9d0038a4 | 0x1bc | 444 | |

Total kseg0_program_mem used for math subroutine:      0x1c60      7264

With 9 math subroutine, each of them has memory usage of 7264/9=807

Exercise 5.4

```
  u3 = u1 & u2; // bitwise AND
9d00224c:   8fc30000    lw    v1,0(s8)
9d002250:   8fc20004    lw    v0,4(s8)
9d002254:   00621024    and   v0,v1,v0
9d002258:   afc20008    sw    v0,8(s8)
```

4 commands

```
  u3 = u1 | u2; // bitwise OR
9d00225c:   8fc30000    lw    v1,0(s8)
9d002260:   8fc20004    lw    v0,4(s8)
9d002264:   00621025    or    v0,v1,v0
```

```
9d002268:     afc20008     sw     v0,8(s8)
```

4 commands

```
  u3 = u2 << 4; // shift left 4 spaces, or multiply by 2^4 = 16
9d00226c:     8fc20004     lw     v0,4(s8)
9d002270:     00021100     sll    v0,v0,0x4
9d002274:     afc20008     sw     v0,8(s8)
```

3 commands

```
  u3 = u1 >> 3; // shift right 3 spaces, or divide by 2^3 = 8
9d002278:     8fc20000     lw     v0,0(s8)
9d00227c:     000210c2     srl    v0,v0,0x3
9d002280:     afc20008     sw     v0,8(s8)
```

3 commands

Exercise 6.1

Polling can only process at a regular interval. Interruption can happen in any given time.

Polling

Pros: simple program with fewer lines of assembly languages compare to interrupt, transmission reliability that takes place at maximum speed

Cons: standby time is shorter than the response time, need to apply another method of transmission

Interrupt

Pros: serves multiple devices, more flexible and efficient which eliminates nested loops, remove ambiguities in the code.

Cons: requirement for more complex hardware which store noncacheable memory for SFRs and loss of time, need to read data sheet to understand SFRs.

Exercise 6.4

(a) An IRQ is generated for an ISR at priority level 4, subpriority level 2 while the CPU is in normal execution, the CPU will attend the ISR and finished the interrupt then return to original task.

(b) IRQ is generated while the CPU is executing a priority level 2, subpriority level 3 ISR, the CPU will attend the level 4 ISR since level 4 is have higher priority than level 2 and stop the priority level 2, subpriority level 3 ISR until it finished.

(c) IRQ is generated while the CPU is executing a priority level 4, subpriority level 0 ISR, the CPU will attend priority level 4, subpriority level 0 ISR first since they have the same priority and we need to compare the subpriority and 2 has higher priority. However, we cannot interrupt the sequence and have to wait until the task is finished.

(d) IRQ is generated while the CPU is executing a priority level 6, subpriority level 0 ISR, the CPU will continue attending the level 6 ISR since it has higher level compare to priority level 4.

Exercise 6.5

(a) If an interrupt generated, the first thing the CPU does is save the contexts to RAM. Then, the CPU restores the contexts back from the RAM into the registers and the CPU reverts to its initial state. This process is called context save and restore.

(b) Shadow register is an extra register of CPU, when called a particular priority level, CPU use the set of registers rather than context save and restore. By using shadow register, it can save processing time.

Exercise 6.8

(a)
```
IEC0SET = 0x100;          // Enabling the Timer2 interrupt
IFS0CLR = 0x100;          // Setting flag status to 0
IPC2CLR = 0x1F;           // Clearing IPC2 Register
IPC2SET = 0x16;           // Setting priority level to 5, sub priority level to 2
```

(b)
```
IEC1SET = 0x8000;         // Enabling the Real-Time Clock and Calendar interrupt
IFS1CLR = 0x8000;         // Setting flag status to 0
IPC8CLR = 0x1F000000;     // Clearing IPC8 Register
IPC8SET = 0x19000000;     // Setting priority level to 6, sub priority level to 1
```

(c)
```
IEC2SET = 0x10;           // Enabling the UART4 Receiver interrupt
IFS2CLR = 0x10;           // Setting flag status to 0
IPC12CLR = 0x1F00;        // Clearing IPC2 Register
IPC12SET = 0x1F00;        // Setting priority level to 7, sub priority level to 3
```

(d)
```
IEC0SET = 0x800;          // Enabling the INT2 external input interrupt
IFS0CLR = 0x800;          // Setting flag status to 0
```

```
    IPC2CLR = 0x1F00000;        // Clearing IPC2 Register
    IPC2SET = 0xE000000;        // Setting priority level to 3, sub priority level to 2
    INTCONSET = 0x8;            // Configuring to trigger on rising edge
```

Exercise 6.9

The code is in 6-9 zip file

Exercise 6.16

The code is in 6-16 zip file

Exercise 6.17

The code is in 6-17 zip file