4357. Embedded Firmware Essentials
Homework #1
Jae Yang Park (jaeyangp@gmail.com)

Source

```
// HW01
// Assigned Dip pin# 29
// port 0, pin 5

#define FIO0DIR0    (*(volatile unsigned int *)(0x2009c000))
#define FIO0PIN0    (*(volatile unsigned int *)(0x2009c014))
#define PIN5_H      ((unsigned char)(1 << 5))
#define PIN5_L      ((unsigned char)(~(1 << 5)))
#define DIR_OUT     ((unsigned char)(~0))

int main()
{
    FIO0DIR0 = DIR_OUT;    // set port 0 output mode

    register unsigned char h, l;
    h = FIO0PIN0 | PIN5_H;
    l = FIO0PIN0 & PIN5_L;

    while (1) {
        FIO0PIN0 = h;      // output 1
        FIO0PIN0 = l;      // output 0
    }
}
```

Compile

```
$ arm-none-eabi-gcc -mcpu=cortex-m3 -mthumb -c hw01.c
$ arm-none-eabi-gcc -mcpu=cortex-m3 -mthumb -O3 -c hw01.c -o hw01_O3.o
$ arm-none-eabi-gcc -mcpu=cortex-m3 -mthumb -Os -c hw01.c -o hw01_Os.o
```

Dump file

```
$ arm-none-eabi-objdump -d hw01.o
```

No optimization

```
     0:       b4b0          push   {r4, r5, r7}
     2:       af00          add    r7, sp, #0
     4:       4b0a          ldr    r3, [pc, #40] ; (30 <main+0x30>)
     6:       22ff          movs   r2, #255      ; 0xff
     8:       601a          str    r2, [r3, #0]
     a:       4b0a          ldr    r3, [pc, #40] ; (34 <main+0x34>)
     c:       681b          ldr    r3, [r3, #0]
     e:       b2db          uxtb   r3, r3
    10:       f043 0320     orr.w  r3, r3, #32
    14:       b2dd          uxtb   r5, r3
    16:       4b07          ldr    r3, [pc, #28] ; (34 <main+0x34>)
    18:       681b          ldr    r3, [r3, #0]
    1a:       b2db          uxtb   r3, r3
    1c:       f023 0320     bic.w  r3, r3, #32
    20:       b2dc          uxtb   r4, r3
    22:       4b04          ldr    r3, [pc, #16] ; (34 <main+0x34>)
    24:       462a          mov    r2, r5
    26:       601a          str    r2, [r3, #0]
    28:       4b02          ldr    r3, [pc, #8]  ; (34 <main+0x34>)
    2a:       4622          mov    r2, r4
    2c:       601a          str    r2, [r3, #0]
    2e:       e7f8          b.n    22 <main+0x22>
    30:       2009c000      .word  0x2009c000
    34:       2009c014      .word  0x2009c014
```

Optimization –O3 (turns on all optimizations specified by –O2)

```
00000000 <main>:
   0:       4a07          ldr    r2, [pc, #28] ; (20 <main+0x20>)
   2:       4908          ldr    r1, [pc, #32] ; (24 <main+0x24>)
   4:       20ff          movs   r0, #255      ; 0xff
   6:       6010          str    r0, [r2, #0]
   8:       680a          ldr    r2, [r1, #0]
   a:       460b          mov    r3, r1
   c:       6809          ldr    r1, [r1, #0]
   e:       f042 0220     orr.w  r2, r2, #32
  12:       b2d2          uxtb   r2, r2
  14:       f001 01df     and.w  r1, r1, #223  ; 0xdf
  18:       601a          str    r2, [r3, #0]
  1a:       6019          str    r1, [r3, #0]
  1c:       e7fc          b.n    18 <main+0x18>
  1e:       bf00          nop
  20:       2009c000      .word  0x2009c000
  24:       2009c014      .word  0x2009c014
```

Optimization –Os (Optimize for size)

```
00000000 <main>:
   0:       4b06          ldr    r3, [pc, #24] ; (1c <main+0x1c>)
   2:       22ff          movs   r2, #255      ; 0xff
   4:       601a          str    r2, [r3, #0]
   6:       3314          adds   r3, #20
   8:       681a          ldr    r2, [r3, #0]
   a:       6819          ldr    r1, [r3, #0]
   c:       f042 0220     orr.w  r2, r2, #32
  10:       b2d2          uxtb   r2, r2
  12:       f001 01df     and.w  r1, r1, #223  ; 0xdf
  16:       601a          str    r2, [r3, #0]
  18:       6019          str    r1, [r3, #0]
  1a:       e7fc          b.n    16 <main+0x16>
  1c:       2009c000      .word  0x2009c000
```

@	Assembly	Machine Code	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Description
0	ldr r3, [pc, #24]	4b06	0	1	0	0	1	0	1	1	0	0	0	0	0	1	1	0	FIO0DIR0
2	movs r2, #255	22ff	0	0	1	0	0	0	1	0	1	1	1	1	1	1	1	1	
4	str r2, [r3, #0]	601a	0	1	1	0	0	0	0	0	0	0	0	1	1	0	1	0	Set port0 output mode
6	adds r3, #20	3314	0	0	1	1	0	0	1	1	0	0	0	1	0	1	0	0	FIO0PIN0
8	ldr r2, [r3, #0]	681a	0	1	1	0	1	0	0	0	0	0	0	1	1	0	1	0	r2 ← FIO0PIN0
a	ldr r1, [r3, #0]	6819	0	1	1	0	1	0	0	0	0	0	0	1	1	0	0	1	r1 ← FIO0PIN0
c	orr.w r2, r2, #32	f042 0220	1	1	1	1	0	0	0	0	0	1	0	0	0	0	1	0	high
			0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	
10	uxtb r2, r2	b2d2	1	0	1	1	0	0	1	0	1	1	0	1	0	0	1	0	one byte from r2
12	and.w r1, r1, #223	f001 01df	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	low
			0	0	0	0	0	0	0	1	1	1	0	1	1	1	1	1	
16	str r2, [r3, #0]	601a	0	1	1	0	0	0	0	0	0	0	0	1	1	0	1	0	Set pin 5 high
18	str r1, [r3, #0]	6019	0	1	1	0	0	0	0	0	0	0	0	1	1	0	0	1	Set pin 5 low
16	b.n 16	e7fc	1	1	1	0	0	1	1	1	1	1	1	1	1	1	0	0	goto 16

Assembly	Op code	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Description
ldr Rd,[pc, #imm8]		0	1	0	0	1	Rd			imm8*								Rd ← pc + #imm8
ldrb Rd, [Rn, #imm5]		0	1	1	1	1	imm5					Rn			Rd			Rd ← Rn + #imm5
str Rd, [Rn, #imm5]		0	1	1	0	0	imm5					Rn			Rd			[Rn + #imm5] ← Rd
strb Rd, [Rn, #imm5]		0	1	1	1	0	imm5					Rn			Rd			[Rn + #imm5] ← Rd
b.n label		1	1	1	0	0	offset_11**											12-bit two's complement

(from ARM Architecture Reference manual ARMv7-M,)

Assembly	Op code																Description
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
ldr Rd,[pc, #imm8]	0	1	0	0	1	Rd			imm8*								Rd ← pc + #imm8
orr.w Rd, Rn, #imm8	1	1	1	1	0	i	0	0	0	1	0	S	Rn				Rd ← Rn OR #imm8
	0	imm3			Rd				imm8								
and.w Rd, Rn, #imm8	1	1	1	1	0	i	0	0	0	0	0	S	Rn				Rd ← Rn AND #imm8
	0	imm3			Rd				imm8								
bic.w Rd, Rn, #imm8	1	1	1	1	0	i	0	0	0	0	1	S	Rn				Rd ← Rn AND (NOT #imm8)
	0	imm3			Rd				imm8								
mov.w Rd, #imm16	1	1	1	1	0	i	0	0	0	1	0	S	1	1	1	1	Rd ← #imm16
	0	imm3			Rd				imm8								
uxtb Rd, Rn	1	0	1	1	0	0	1	0	1	1	Rn			Rd			Rd ← extract 8 bits from Rn

Note:

`imm8*` : The value specified by `#imm8` is a full 10-bit address, but must always be word-aligned (ie with bits 1:0 set to 0), since the assembler places `#imm8 >> 2` in field Word8.

`offset_11**:` The address specified by label is a full 12-bit two's complement address, but must be always halfword aligned (i.e, bit 0 set to 0), since the assembler places `label >> 1` in the `offset_11` field.

Difference between GCC Optimization options

`` `-O0' ``
No optimization (the default); generates unoptimized code but has the fastest compilation time.

`` `-O1' ``
Moderate optimization; optimizes reasonably well but does not degrade compilation time significantly.

`` `-O2' ``
Full optimization; generates highly optimized code and has the slowest compilation time.

`` `-O3' ``
Full optimization as in `` `-O2' ``; also uses more aggressive automatic inlining of subprograms within a unit (Inlining of Subprograms) and attempts to vectorize loops.

`` `-Os' ``
Optimize space usage (code and data) of resulting program.

Result: The size of code generated is reduced with the optimization option, and the registers uses are more efficient.
In the modified code, the size of loop is obviously reduced.

Total spent hours: 8 hours

- Coding, compile and objdump: 0.5
- Modification and trial with different optimization options: 5.5
- Finding instruction sets: 1
- Document: 1