CEG411 Microprocessor Based System Design

- What is this course about?
- Computer : Processor, Memory, I/O
- Microprocessor versus microcontroller
- EVB (Evaluation Board): Axiom CSM12C32
- Chip: Freescale (formerly, a Motorola division) MC9S12C32
- 68HC12 versus 68HCS12

Lab 1 Preparation

- Monitor, MON12, Monitor commands (Sec. 3.7.3) MD, MM, LOAD, CALL
- Address versus data
- Hexadecimal number system and Memory map
- C, Assembly, Machine Code
- ICC12 and as12
- Step by step instructions
- S record file, list file, map file

Sample C Programs

- C review (Chapter 5, sample_c.c)
  - main( ), function call
  - data types (char, unsigned char, int, short, long, float)
  - Condition check (true, false, if, else)
  - for loop, while loop
  - putchar(), puts(), printf()
- tone.c : timer, output compare (Sec 8.6)
- switch_LED.c : simple parallel port usage

Simple Parallel Port Usage

- Let Y be a generic register which can be PORTA, PORTB, PORTE, PTAD, PTT, PTP, PTS, PTM, DDRA, DDRB, and so on.
- Contents of Y: $Y_7\ Y_6\ Y_5\ Y_4\ Y_3\ Y_2\ Y_1\ Y_0$
- Output (Use Y_1 as an example)
  - Set bits : Y |= 0x02;
  - Clear bits : Y &= ~0x02; or Y &= 0xFD;
  - Toggle bits : Y ^= 0x02;
- Input
  - Check if set : if(Y & 0x02)
  - Check if clear : if(!(Y & 0x02))
  - Wait until set : while(!(Y & 0x02))
  - Wait until clear: while(Y & 0x02)

Interrupt Programming

- Interrupt versus polling
- Interrupt programming
  - Write an ISR (interrupt service routine)
  - Register the ISR
  - Enable ISR (locally and globally)
- tone_interrupt.c example
- On EVB, pressing the reset button clears the MON12 (user) interrupt vector table contents
- Chapter 6

ADC (Chapter 10)

- ADC basics, a 2-bit ADC example, Sec. 10.4.1, Example 10.1
- ADC Internal: Successive Approximate Method
- Signal Conditioning Circuits (based on OP Amp)
- 6812 ADC Programming
  - adc.c
  - adc_scan.c

Assembly Programming

- Why assembly programming?
  - Understand computer/C internal operations
  - More efficient coding
- CPU Registers (Sec. 1.4)

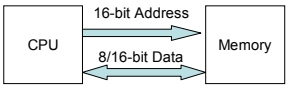
  D (A:B) : accumulator; X, Y: index registers;

  SP : Stack Pointer; PC: Program Counter

  CCR : Condition Code Register

  S X H I N Z V C

- Memory Addressing (Sec. 1.5)



  - Big-Endian versus Little-Endian
  - Memory Mapped I/O
- A sample assembly program (tonevb.asm)
  - Label, Opcode, Operands, Comments
  - Assembler directives (e.g., ORG)

- Load/Store Instructions (Sec. 1.8.1) and Addressing Modes (Sec. 1.6)
- Examples
  - LDAA #$55 immediate mode
  - LDAA #55 immediate mode
  - LDD #$0F20 immediate mode
  - LDAA $55 direct mode
  - LDAA $0F20 extended mode
  - LDD $0F20 extended mode
  - LDAB 3,X indexed mode
- LDX, LDY, LEAS, LEAX, LEAY
  - LEAS -4,SP : SP ← (SP)-4
    (to allocate space for local variables for subroutines)
- STAA, STAB, STD, STS, STX, STY

Example:

```
int m, n;  // assume m is at $3000
           // assume n is at $3002
m = n + 5;

LDD   $3002
ADDD #5
STD   $3000
```

How about char instead of int?

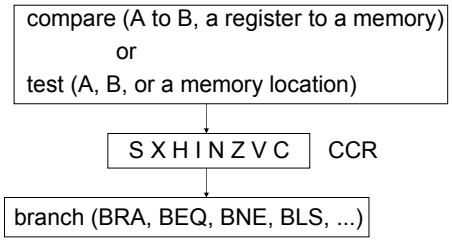
- More instructions (Sec. 1.8)
  - Transfer (register to register) and Exchange (swap registers):

    TAB, TAP, TBA, ...; EXG, XGDX, XGDY
  - Move (memory to memory): MOVB, MOVW
  - Add and Subtract (always involves CPU registers)
    - Register + Register : ABA, ABX, ABY
    - Register + Memory : ADCA, ADCB, ADDA, ADDB, ADDD
    - Register – Register : SBA
    - Register – Memory : SBCA, SBCB, SUBA, SUBB, SUBD

- Assembler Directives (Sec. 2.3)
  - ORG (Origin), EQU (Equate)
  - Specify Constants: fcb (Form Constant Byte, or db), fcw (Form Constant Word, or dw), fcc (Form Constant Character, String), fill
  - Reserve Space: rmb (reserve memory byte), rmw (reserve memory word)
- Examples

```
        ORG   $3800
array   fcb   $11,$22,$33,'5,25,100
buf     rmb   20
msg     fcc   "Hello World"
tmp     fill  $11,20
```

- Program Loops (Sec. 2.6)
  - Loop constructs
  - Branch conditions/instructions



- Examples 2.14 (p. 57), 2.15
  - Draw flowchart
  - Revise flowchart
- Example 2.17
  - BRCLR oprand,mask,label
    branch to label if the bit(s) is (are) clear
    while(!(TFLG1 & 0x01)); // wait until flag is set
    // the above C statement is equivalent to
    here BRCLR $004E,$01,here
    Note that $004E is the address of TFLAG1
  - BRSET oprand,mask,label

Subroutine Calls

- Stack, Stack Pointer, Push, Pull
  - PSHA, PSHB, PSHC, PSHD, PSHX, PSHY
    : Decrement (SP) first, then write data
  - PULA, PULB, PULC, PULD, PULX, PULY
    : Read data first, then increase (SP)
- Example:

```
LDS    #$3E00
LDAA   #$3B
LDD    #$302F
PSHA
PSHD
```

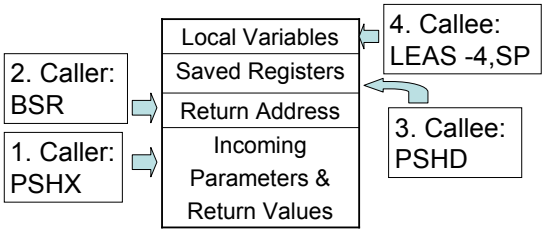
- Ex. 4.1

- BSR, JSR, RTS, Caller, Callee (Sec. 4.6)

```
       BSR   SORT
       ........
SORT   ........
       ........
       RTS
```

- Parameter Passing, Result Returning, and Allocation of Local Variables (Sec. 4.7)
  - Using registers versus using stack

- Stack Frames (Sec. 4.8), Ex. 4.10



- Example C versus Assembly

```
void main()
{
  int a, b;
  a = sum(b, 3);
}

int sum(int x, int y)
{
   int s;
   s = x + y;
   return s;
}
```

```
LDD 5,sp      *** b
PSHD
LDD #3
PSHD
LEAS -2,sp  *** result
JSR Sum
....

Sum:
      PSH...    *** save reg
      LEAS -2,sp *** s
      ......
```

More Timer Functions

- Timer Overflow (p. 206 and p. 293): TSCR2 bit 7 (TOI), TFLAG2 bit 7(TOF)
- Input Capture (Sec. 8.5) and Examples
- Real-Time Interrupt (Sec. 8.7):
  CRGINT bit 7 (RTIE), CRGFLG bit 7 (RTIF)
  m = RTICTL bits 6-4; n = RTICTL bits 3-0
  OSCCLK (16MHz on EVB)/[(m+1)$2^{(n+9)}$]
- Pulse Accumulator: a 16-bit counter
- Pulse Width Modulation (PWM) Function

- Parallel Ports, I/O Synchronization and Handshaking (Sec. 7.4)
- Serial Interface (Chapter 9): SCI vs. SPI
  - RS-232, Start-bit, Stop-bit, Parity
- SCI
  - SCIBDH, SCIBDL
  - SCICR1, SCICR2
  - SCISR1, SCISR2
  - SCIDRH, SCIDRL
- SPI (p. 391)