University of British Columbia
Electrical and Computer Engineering
Digital Systems and Microcomputers CPEN312

Lecture 13a: Introduction to 8051 Assembly I

Dr. Jesús Calviño-Fraga P.Eng.
Department of Electrical and Computer Engineering, UBC
Office: KAIS 3024
E-mail: jesusc@ece.ubc.ca
Phone: (604)-827-5387

March 7, 2017

Objectives

- Get acquainted to the 8051 instruction set.
- Understand how an assembler compiler is used and the files it generates.
- Assemble mnemonics by hand.
- Disassemble machine code by hand.
- Calculate machine code execution time.
- Write simple assembly programs.
- Load machine code by hand.

8051 Instruction Set

1. Arithmetic Operations
2. Logical Operations
3. Data Transfer
4. Boolean Variable Manipulation
5. Program Branching and Machine Control

8051 Instruction Set

- The instructions are all described in chapter 2 of “MCS-51 Microcontroller Family User’s Guide”.
- The number of cycles per instruction for the CV-8052 are listed in the one page handout distributed:
  - Side 1: Instructions sorted by opcode number
  - Side 2: Instructions sorted by mnemonic name.

Assembly Language Instruction

- **An Assembly language instruction consists of four fields:**

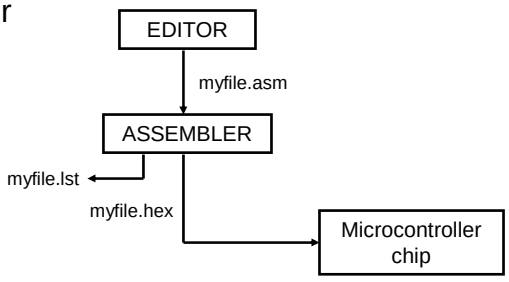
  [label :] mnemonic [operands] [;comment]

- **Examples:**

```
L1:  MOV A, #'*' ; Load Acc with ASCII for '*'
     NOP
     ; We can just have comments!
L3: ; We can also place labels anywhere
     DJNZ R0, L3
```


Writing and compiling an assembly language program

Using A51 assembler
by yours truly!



Hand writing, compiling, and loading an assembly program

By hand: Good for educational purposes, but very, very impractical for real applications!

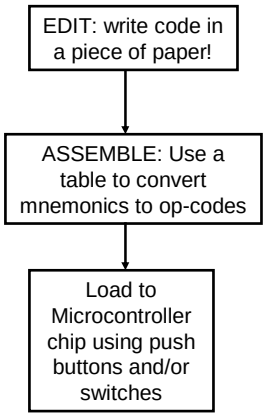


Table of mnemonics is available as an Excel spreadsheet in the course web page: *'CV-8052_opcodes.xls'*

Exercise: Assembling Machine Code by Hand

Assemble by hand the following piece of machine code:

```
E9    MOV A, R1     ; Low 8-bits first number
2B    ADD A, R3     ; Add to low 8-bits second number
FD    MOV R5, A     ; Save result-low to R5
E8    MOV A, R0     ; High 8-bits first number
3A    ADDC A, R2    ; Add  with carry to high 8-bits second number
FC    MOV R4, A     ; Save result-high to R4
```

Answer: E9 2B FD E8 3A FC

This type of question often appears in exams

Disassembling

- In microcomputer jargon, disassembling is the process of getting the source code from the machine code.
- Many debuggers disassemble machine code to trace program execution.

Exercise: Disassembling by hand

- What is the value of R2 after executing this piece of 8051 machine code?
  C3 74 88 94 10 FA

```
C3          CLR C          ; Set carry to zero
74 88       MOV A, #88H    ; Load Acc. With 88H
94 10       SUBB A, #10H   ; 88H-10H=78H
FA          MOV R2, A      ; R2=78H
```

Answer!

This type of question often appears in exams

Number representation

- Decimal (default): 2957 or 2957D
- Binary: 101110001101B
- Octal: 5615o or 5615Q
- Hexadecimal: 0B8DH, 0b8dh, 0x0B8D, 0x0b8d

The maximum number that can be input in any radix is 65535 which corresponds to 16 bits. If we have a 'equ' formula, the result should fit in 16 bits.

Machine cycles for the 8051 Instructions

- The microcontroller takes a certain number of clock cycles to execute an instruction.
- These clock cycles are referred to as *machine cycles.*
- In the 8051 family, the length of the machine cycle depends on the frequency of the crystal oscillator.
- The frequency of the crystal connected to the 8051 family can range from 32kHz to over 1000 MHz.
- In the original 8051, one machine cycle lasts 12 oscillator periods. Therefore, to calculate the machine cycle for the original 8051, we take 1/12 of the crystal frequency, then take its inverse.
- For newer 8051 derivatives, one machine cycle can take 12, 6, 4, 2, or 1 oscillator period(s).
- Warning: The same instruction can take a different number of machine cycles in 8051's fabricated by different manufacturers:

Machine cycles for the 8051 Instructions

Instruction	Intel 8051	CPEN312 CV-8052	Atmel AT89LP52	Silicon Labs C8051F38x
MOV R3, #value	1	2	2	2
DEC Rx	1	1	1	1
DJNZ	2	2/3	2	2/4
LJMP	2	3	3	5
SJMP	2	2	2	4
NOP	1	1	1	1
MUL AB	4	1	2	4

Machine cycles for the CV_8052

- Available in Excel format in the course web page.
- The CV_8052 takes one clock per cycle!
- The clock of the CV_8052 is 33.333333MHz. Therefore, each cycle is 1/33.333333MHz=30ns.
- The opcode table for ALL 8051 compatible microcontrollers is the same. The number of bytes per instruction is the same. The number of cycles per instruction MAY NOT BE the same.

Example 1: Instruction Timing for the original 8051 from Intel

- How long will it take this program to run in a standard 8051 (12 clocks per cycle) with a 12MHz crystal? Use the cycles per instruction of Appendix G in the textbook.

CODE: C3 74 88 94 10 FA

```
C3          CLR C          ; Set carry to zero (1 cycle)
74 88       MOV A, #88H    ; Load Acc. With 88H (1 cycle)
94 10       SUBB A, #10H   ; 88H-10H=78H (1 cycle)
FA          MOV R2, A      ; R2=78H (1 cycle)
```

1 cycle=12/12MHz=1µs, therefore 4 cycles take 4µs

Example 2: Instruction Timing for the CV_8052

- How long will it take this program to run in CV_8052 (1 clock per cycle) with a 33.3333 MHz clock?

CODE: C3 74 88 94 10 FA

```
C3          CLR C          ; Set carry to zero (1 cycle)
74 88       MOV A, #88H    ; Load Acc. With 88H (2 cycles)
94 10       SUBB A, #10H   ; 88H-10H=78H (2 cycles)
FA          MOV R2, A      ; R2=78H (1 cycle)
```

1 cycle=1/33.33MHz=30ns, therefore 6 cycles take 0.18µs

This type of question often appears in exams

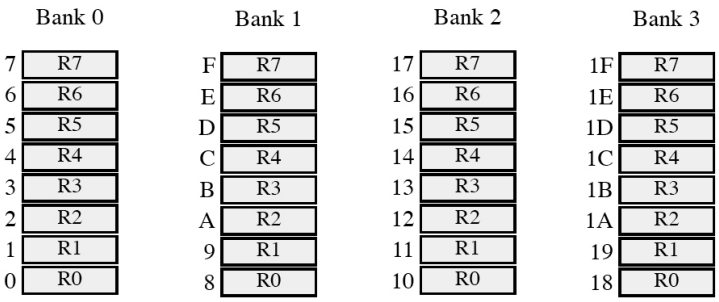
8051 Registers

- The 8051 is LOADED with registers, all of them (but one) are 8-bit:
  - 32 general purpose registers arranged in 4 banks of 8 register each: R0 to R7. R0 and R1 can be used as “index” registers for indirect access.
  - N (depends on the derivative, but usually 27 or more) Special Function Registers or SFR.
- All the hardware resources of the 8051 are accessed through SFRs. More on to this later.

8051 Registers

- Just a few registers are related to ALU operations:
  - Accumulator (Acc).
  - Register B.
  - Data Pointer DPTR (two eight bit registers put together: DPH and DPL)
  - Stack Pointer SP.
  - Program Counter PC.
  - Program Status Word PSW.

Registers R0 to R7



Accumulator

- It is the most versatile of all the registers. All the ALU operations place the result in the accumulator.
- Curiously, the accumulator is also a Special Function Register (SFR) in the 8051.
- Bit Addressable. We can use the setb, clr, jb, etc. with the accumulator bits: setb acc.3
- Many opcodes accept only the accumulator as operand.
- Example usage:
  - MOV a, #20H ; Load accumulator with 20H
  - INC a ; Add one to accumulator
  - SWAP A ; Swap bits 0 to 3 with bits 4 to 7

Register B

- Used together with the Accumulator to perform 8-bit multiplication/division operations.
- Also bit addressable.
- Example usage:
  - MOV A, #140
  - MOV B, #150
  - MUL AB ; After this inst. A=08H, B=52H

Data Pointer DPTR

- Formed by two 8-bit SFRs: DPH and DPL.
- Together with some especial opcodes, it is used to access CODE and XRAM memory.
- This is the only 16-bit register in the 8051.
- There is a dedicated opcode to increment it! Unfortunately there is no opcode to decrement it!
- Example:
  - MOV DPTR, #0AAAAH
  - INC DPTR ; DPL=0ABH, DPH=0AAH

How to decrement the DPTR

```
        mov a, dpl
        jnz skip_dph
        dec dph
skip_dph:
        dec dpl
```

Exercise: modify the code above so that the value of the accumulator remains unchanged.

Stack Pointer

- Need to be set for the 'CALL'/'RET' instructions to work properly.
- The stack will be cover with more detail in future lectures.
- For now, at the beginning of your code place the following instruction:
  - MOV SP, #7FH ; Set sp to beginning of IDATA!

Program Counter

- We can not access this 16-bit register directly. Increments by one, two, or three bytes (one for most instructions), depending on the opcode length.
- We can only use the “jump”, “call”, or “return” instructions to change the program counter.
- Example:
  - LJMP 34AAH ; Set PC to 34AAH

Program Status Word

- It is where the ALU are stored:

CY	AC	FO	RS1	RS0	OV	--	P

CY	Carry flag
AC	Auxiliary Carry flag (For BCD Operations)
FO	Flag 0 (Available to the user for General Purpose)
RS1, RS0	Register bank select bits.
OV	Overflow flag
P	Parity flag

Example: JC 8 ; If the carry is set jump 8 bytes ahead

Instructions that modify the 'Flag' bits in the PSW:

Instruction	CY	OV	AC
ADD	X	X	X
ADDC	X	X	X
SUBB	X	X	X
MUL	0	X	
DIV	0	X	
DA	X		
RRC	X		
RLC	X		
SETB C	1		
CLR C	0		
CPL C	X		
ANL C, bit	X		
ANL C, /bit	X		
ORL C, bit	X		
ORL C, /bit	X		
MOV C, bit	X		
CJNE	X		

Example 3

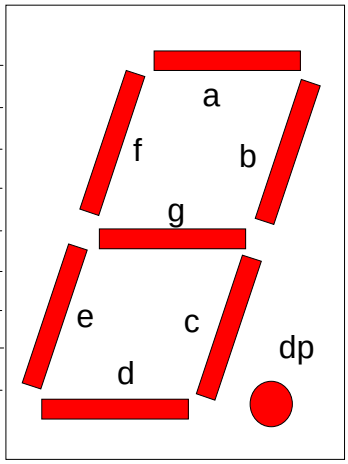
- Write an assembly program that writes 'JESUS' to the 7-segment displays HEX4 to HEX0. Compile and load the program manually. The Special Function Register (SFR) addresses of HEX4 to HEX0 are:

  HEX0: 91H
  HEX1: 92H
  HEX2: 93H
  HEX3: 94H
  HEX4: 8EH

  The segments of each display are mapped as : -gfedcba, 'a' is the least significant bit.

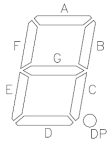
7-Segment Displays in DE0-CV board

	CA
bit 0	a
bit 1	b
bit 2	c
bit 3	d
bit 4	e
bit 5	f
bit 6	g
NC	dp



Disp.	SFR
HEX0	91H
HEX1	92H
HEX2	93H
HEX3	94H
HEX4	8EH
HEX5	8FH

Segments turn on with zero.



7-segment constants

- The bit pattern for each letter is:

Letter	g	f	e	d	c	b	a	Hex
'J'	1	1	0	0	0	0	1	61H
'E'	0	0	0	0	1	1	0	06H
'S'	0	0	1	0	0	1	0	12H
'U'	1	0	0	0	0	0	1	41H
'S'	0	0	1	0	0	1	0	12H

Exercise: Obtain the bit patterns for numbers 0 to 9

The assembly code

Address	Machine Code	Instruction
0000	75 8E 61	mov 8EH, #61H
0003	75 94 06	mov 94H, #06H
0006	75 93 12	mov 93H, #12H
0009	75 92 41	mov 92H, #41H
000C	75 91 12	mov 91H, #12H
000F	80 FE	sjmp $

Loading the Code by Hand

- First load the POF file CV_8052.POF to the Altera DE0-CV board (you need to do this only once):
  - Set SW10 to the 'PROG' position
  - Turn the board on. Connect to computer and load Quartus.
  - Click the 'programmer' icon in Quartus and send the above POF file to the DE0-CV board.
  - Set SW10 to the 'RUN' position.
- Second: We need to activate 'manual load' in the CV_8052 soft processor by pressing and releasing FPGA_RESET while KEY2 is pressed.

Manual load in the CV_8052

- HEX5 to HEX2 show the memory address while HEX1 and HEX0 show the current data at that memory address.
- SW8 selects address LOW entry.
- SW9 selects address HIGH entry.
- SW0 to SW7 value to enter.
- KEY3 increments the address by one.
- KEY2 decrements the address by one.
- KEY1 enters data.
- KEY0 and KEY1 at the same time stores changes to flash.

Example 4 (time permitting)

- Write a program that turns LEDR0 on/off every 0.5s. Compile and load the program manually. The bit address of LEDR0 is E8H.

Big delay: loops inside loops

```
main_loop:
    mov R2, #90  ; 90 is 5AH
L3: mov R1, #250 ; 250 is FAH
L2: mov R0, #250
L1: djnz R0, L1  ; 3 machine cycles-> 3*30ns*250=22.5us
    djnz R1, L2  ; 22.5us*250=5.625ms
    djnz R2, L3  ; 5.625ms*90=0.506s (approximately)
    cpl 0E8H     ; Bit address of LEDR0 is E8H
    sjmp main_loop
```

One possible solution! The same result can be achieved using many other valid programs.

Program compiled by hand

```
                main_loop:
0000  7A 5A         mov R2, #90
0002  79 FA     L3: mov R1, #250
0004  78 FA     L2: mov R0, #250
0006  D8 FE     L1: djnz R0, L1
0008  D9 FA         djnz R1, L2
000A  DA F6         djnz R2, L3
000C  B2 E8         cpl 0E8H
000E  80 F0         sjmp main_loop
```

Exercises

- The machine code below runs in a CV-8052 processor that takes 1 clock per cycle with a 33.3333MHz clock. Find how long it takes this code to execute. (Warning: there is a loop you have to consider!)

  7E 08 0E 00 8E 80 00 00 DE FA

- Assemble by hand (both op-codes and operands) the program below.

```
MOV R7, #75H
MOV A, R7
ANL A, #0FH
ORL A, #30H
MOV R0, A
MOV A, R7
SWAP A
ANL A, #0FH
ORL A, #30H
MOV R1, A
```

- Modify the programs of examples 3 and 4 so that they turn off all unused LEDs by writing zero to them. The SFR addresses for the LEDS are: LEDR0-7=E8H, LEDR8-9=95H.