Microprocessor HW#1

107061112　王昊文

EE2401 微算機系統Fall 2019

HW#1 (8051 Hardware, Instruction Set) (9/30/2019)

Due date: 10/17/2019. Severe penalty will be given to late homework.

Note:

(a) The homework will be graded based on your **documentation** and **demonstration**.

(b) For all **(Software Design)** problems, you are required to use**MCU8051IDE**simulators to simulate and verify your programs.

(c) You are required to **type** your homework (first the problem then your solution) by using a **word processor** and submit in .doc (or .docx) format under a filename **EE2401f19-hw1-student_no-vn.doc(or .docx)**, where **student_no** is your student number, e.g., **107061xxx** and **vn** is your version number, e.g., **v3**. You should **upload your .doc file** in **iLMS** by the specified deadline whenever you have a newer version. Follow the iLMS upload homework process to upload your file.

(d) The homework will be graded based on your **latest version**. Old version(s) will be discarded.

(e) Each homework assignment will have full score of 100 points. 5 points will be deducted if you do not comply with the naming convention. **Severe grade penalty will be given to late homework and** 抄襲. 20 points will be taken off per day after deadline until zero.

(f) Please treat the above requirements as a kind of training in writing a decent homework report. If you have any problem regarding this homework, please feel free to consult with TA or me. If you think the time is too short to accomplish this homework, please let me know in class.

I. Basic concepts & 8051 instructions: (80%)

1. (15%)

Explain in as much detail as you can the following terms:

(a) Clock cycle, machine cycle, and instruction cycle.

Answer:

Clock cycle:

A **clock cycle** is generated by an on-chip oscillator driven by an external quartz crystal, which typically has a frequency of 12MHZ. These oscillating signals will form a discrete wave, and the time to generate a wavelength long of the discrete wave is called a clock cycle. These clock cycle form the basis of a machine’s timing and synchronization, and each operation performed is in step with these cycles.

Machine cycle:

A **machine cycle** consists of the steps that a computer's processor executes whenever it receives a machine language instruction. Take the 8051 as our

example. The 8051 microprocessor requires two clock cycles to perform a discrete operation, and the duration of two clock cycles is called a state. However, in order to fully perform an instruction, the 8051 requires a minimum of 6 states to perform. The duration of six states, or in other words, 12 clock cycles, is called a machine cycle. A more complex instruction will require more machine cycles to perform.

Instruction cycle:

An **instruction cycle** is the period for executing an instruction. Different instruction may have different time length, usually comes in the unit of machine cycle.

(b) Opcode, operand, mnemonics.

Answer:

Opcode:

An opcode is the portion of a machine language instruction that specifies the operation to be operated. It is a binary number for the computer to read.

Operand:

An **operand**, depending on the language architecture and the opcode before the operand, the operands may be register values, values in the stack, other memory values, I/O ports, etc…… An operand is used to perform operations, including arithmetic, data moving, logical operations and program control.

Mnemonics:

Assembly language replaces the binary opcode with a “Mnemonics” to make programmer easier to facilitate programming. It enhances the readability of assembly code. For example, the code "10110011" in 8051 is to represent arithmetic addition, it might be represented in assembly language by the mnemonic "ADD."

(c) The addressing modes of 8051.

Answer:

There are seven addressing modes in 8051. They are:

(1) Immediate Addressing:
Assigns a constant source into an accumulator.

(2) Direct Addressing
Access any on-chip variable or hardware registers and assign the data in it to an accumulator.

(3) Register Addressing:
Access the working registers “R0-R7” and assign the data in it to the accumulator

(4) Indirect Addressing:
Using R0, R1 as pointers to indicate where the data is, and assign the data in the destination to the accumulator.

(5) Indexed Addressing: Similar to indirect addressing, we can assign the destination to the variable DDRT, in order to jump to the destination, we want, but without using a working register.

(6) Relative Addressing: Used only with certain jump instruction such as JMP and CALL.

(7) Absolute Addressing: Used only with ACALL and AJMP 2-byte instruction. Allow branching within current 2K page.
(8) Long Addressing: Used only with LCALL and LJMP 3-byte instruction.

(d) The instruction types and all the instructions in each type of 8051.

Data Transfer	Arithmetic	Logical	Boolean	Program Branching
MOV	ADD	ANL	CLR	LJMP
MOVC	ADDC	ORL	SETB	AJMP
MOVX	SUBB	XRL	MOV	SJMP
PUSH	INC	CLR	JC	JZ
POP	DEC	CPL	JNC	JNZ
XCH	DIV	RLC	JB	CJNE
	DA a	PR	JBC	NOP
		PRC	ANL	LCALL
		SWAP	ORL	ACALL
			CPL	RET
				RETI
				JMP

(e) Subroutine call, return and stack.
Answer:

Subroutine call:

When performing a **subroutine call**, you are doing what it literally means, calling a subroutine can execute a sequence of program instructions that performs specific tasks, which usually are packaged as a unit. In C Programming subroutines are usually defined in libraries, it may be a procedure, a function or a method. In 8051 programming, The instruction “CALL” can help you jump to the subroutine you desire.

Return:

When you are trying to “**return**” something while programming, there might be some different meanings. While doing assembly languages, when you “return”, you are returning from a subroutine, it will return to the place where you called the subroutine; On the other hand, while doing software programming, in addition to returning a subroutine, it can also return a value whatever the programmer assigns. It may be a character, an integer, or even a function etc……

Stack:

The stack area is an area of RAM allocated to hold temporarily datas. The stack is responsible of reminding the order. In 8051, it should follow the LIFO(Last in first out) rule. The stack pointer (SP) is an 8-bit register at address 81H. It contains the address of the data item currently on the top of the stack. It is initialized as 07H in 8051.

2. (5%)
What is the content of accumulator A after the following instruction sequence executes?

```
MOV     A,      #79H
MOV     50H,    #2FH
MOV     R0,     #50H
XCHD    A,      @R0
```

```
MOV     A,      #79H    ; 79H is stored into accumulator A
MOV     50H,    #2FH    ; 2FH is stored into register 50H
MOV     R0,     #50H    ; 50H is stored into register R0
XCHD    A,      @R0
```

// R0 is pointing to register 50H, 2FH is stored inside of register 50H, now exchange the low-order nibble of the data(bits 0-3) in the accumulator A(0111 1001B) and register 50H(0010 1111B)

A(0111 1001B) -> (0111 1111B)
50H(0010 1111B) -> (0010 1001B)

3. (10%)
Upon reset, the initial value of stack pointer sp is 07H.

(a) What is the value of stack point sp after a series of 3 PUSHes and 1 POP instructions?

Answer:

During every PUSH instruction, it will increase the stack pointer; on the other hand, it will decrease the stack pointer for every POP instruction. Hence, if you execute 3PUSHes and 1 POP instructions, it will increment the initial value by 2, so the final value for the stack pointer is 09H.

(b) What are issues related to stack and register bank 1?
Answer:

- Bank 1 uses the same RAM space as the stack.
- Stack pointer is incremented or decremented according to the push or pop instruction.
- If the stack pointer is decremented it uses locations 7, 6, 5… which belong to register bank 0.
- If a given program uses R1 then stack is provided new memory location.
- The push instruction may also take stack to location 0 i.e.it will run out of space.

4. (5%)

In the short instruction sequence below, what is the instruction executed *after* the `jz Target` instruction.

```
mov r0, #1
mov A,  #2
dec r0
```

```
          jz Target
          mov A,  #3
Target:   mov r0, #7
```

Answer:

Let's take a closer look at what is after jz Target.

```
          jz Target
          mov A, #3
Target:   mov r0, #7
```

To determine whether it will jump, we have to look at what is stored inside accumulator A. Since the decimal number 2 is stored inside A, the program will continue to execute the next line, so the decimal number 3 will be stored inside accumulator A, and then the decimal number 7 will be stored inside register r0.

5. (5%)

Find the execution time of the following sequence. Assume the code is running on an 8051 with a 12MHz crystal. Show all work.

```
          mov    a, #50h
Loop:     dec    a
          cjne   a, #46h, Loop
```

In the case of 8051, a machine cycle is 12 clock cycles. So a machine cycle is

$$\frac{1}{12M} \times 12 = 10^{-6} = 1\mu s$$

MOV requires 1 machine cycle, DEC requires 1 machine cycle, and CJNE requires 2 machine cycles.

Outside the loop, it requires 1 machine cycle, which is the MOV instruction. Inside the loop, the loop will execute 10 times (50H – 10D = 46H), so the execution time is (1 + 2) * 10 = 30.

So the total time to execute the program is

$$1 + 30 = 31$$

It requires $\mathbf{31\mu s}$ to execute the program.

6.(5%)

Write a logical instruction sequence which does the same thing as the following bit instructions:

```
          setb   c
          mov    P1.5,  c
```

```
Answer:
          CLR    c             ; initialize c
          CPL    c             ; setb c
          ORL    P1,    #20H   ; make p1.5 high
```

It is quite trivial using CLP and CPL to set c = 1. Since the bitwise OR logic is only for the carry flag, it is hard to use bitwise OR logic set P1.5 high. Hence by using byte operation, that is using the OR logic on Port 1 with 0010 0000B, we can set P1.5 into high.

Note that it has the same results of what is required, but not actually as same as what happened inside the CPU.

7.(5%)
Write a single assembly language instruction that copies the bit at 95H in internal memory to the carry flag.

Answer:

```
MOV     C,      #95H
```

8. (10%)
Assume the instruction:

AJMP Target

is in code memory at address 4FE0H and 4FE1H, and the label Target corresponds to an instruction at address xxxxH. What are the maximum and minimum values of xxxxH? What are their corresponding hexadecimal machine-language bytes of this instruction for these two cases?

Answer:

4FE0H: 0100 1111 1110 0000 (inside the 01001 page)

Maximum:

4FFFH: 0100 1111 1111 1111, machine language byte is 11101001 11111111

Minimum:

4800H: 0100 1000 0000 0000, machine language byte is 00001001 00000000

9. (10%)
For the 8051 instruction: CJNE R5, #100, LOOP, assuming that the label LOOP refers to an instruction residing 115 bytes before this CJNE instruction,

(a) What is the opcode for this instruction?

Answer:

BDH

(b) How many bytes long is this instruction?

Answer:

3 bytes.

(c) Explain the purpose of each byte of this instruction.

Answer:

The instruction will be encoded as

10111rrr dddddddd eeeeeeee

The first byte consists of the opcode and the number of the register. The second byte is the value of the data, and the last byte is the destination of the label LOOP, also known as the relative address.

(d) How many machine cycles are required to execute this instruction?

Answer:

The instruction CJNE requires **2 machine cycles** to execute.

(e) If an 8051 is operating from a 12MHz crystal, how long does this instruction take to execute?

Answer:

We have calculated that the machine cycle time of a microprocessor with a 12MHz clock is $1\mu s$. Hence the instruction requires two machine cycle so it is $\boldsymbol{2\mu s}$

10.(10%)

An 8051 subroutine is shown below:

```
SUB:    MOV   R0,   #30H
        MOV   R7,   #40H
LOOP:   MOV   @R0,  #0
        INC   R0
        DJNZ  R7,   LOOP
        RET
```

(a) What does this subroutine do?

Answer:

This subroutine is executed like a for loop in C. The loop index will be stored in register R7, which is given as 0x40. The loop will simply keep increasing the value stored in register R0, and also clear the data where the address is stored inside register, which is initially given as 0x30, so the memory after 0x30 will be cleared one at a time as R7 keep decreasing, until R7 counts to zero, then the subroutine will return to where it is called.

(b) In how many machine cycles does each instruction execute?

Answer:

MOV requires 1 machine cycle.
INC requires 1 machine cycle.
DJNZ requires 2 machine cycles.
RET requires 2 machine cycles.

Calculating the machine cycles inside the subroutine and outside the loop, There are two MOV instruction, that is total of 2 machine cycles.

Inside the loop, R7 is treated as the loop index. Hence the loop will execute 40H times, it is 64 times in decimal. There is a total of 4 machine cycles per time, so there will be 64 * 4 = 256 machine cycles. Also, the RET instruction takes 2 machine cycles, so the subroutine requires 2 + 256 + 2 = **260** machine cycles.

(c) How many bytes long is each instruction?

Answer:

MOV Rn, #data is 2 bytes long.
MOV @Ri #data is 2 bytes long.
INC Rn is 1 byte long.
DJNZ Rn rel is 2 bytes long
RET is 1 byte long.

(d) Convert the subroutine to machine language.

```
SUB:    MOV   R0,   #30H   //   01111000   00110000
        MOV   R7,   #40H   //   01111111   01000000
LOOP:   MOV   @R0,  #0     //   01110110   00000000
        INC   R0           //   00001000
        DJNZ  R7,   LOOP   //   11011111   eeeeeeee
        RET                //   00100010
```

Note that eeeeeeee stands for the relative address of LOOP.

(e) How long does this subroutine take to execute? (Assume 12-MHz operation.)
Since it requires 260 machine cycles, it will take a total of $\boldsymbol{260\mu s}$

II.Basic hardware/software design (20%)

1. (20%)

Design an 8032-based single board computer:

Ext. code memory:	0000H-3FFFH: 16KB EPROM chip 1
	4000H-7FFFH: 16KB EPROM chip2
	8000H-FFFFH: 32kB SRAM
Ext. data memory:	8000H-FFFFH: 32kB SRAM
	0000H-7FFFH: 8255 I/O expansion

by using a computer tool (e.g., DesignWork Lite, OrCAD, Protel, etc.) for drawing the schematic.

(a) Search the internet for these memory ICs and give a brief explanation of them.
Answer:

8032 microprocessor:

First let's talk about 8051 and 8052. 8051 is like the older version of 8052, 8052 has 8k on chip code memory, and 256 Bytes on chip data memory, which the storage is twice larger than 8051. The 8032 is architecturally similar to 8052, but without an internal ROM, it's like the romless version of 8052.

16KB EPROM:

Using the IC **27128** as our 16KB EPROM. The full name for EPROM is Erasable Electrically Programmable Read Only Memory, the 27128 provides microprocessor-based systems extensive storage capacity for large portions of operating system and application software. Its 100ns access time provides high speed operation with high speed CPUs. The 27128 is configured in the standard EPROM pinout which provides an easy upgrade path for systems which are currently using standard EPROMs.

32KB SRAM:

We use the IC **62256** as our 32KB SRAM module. The 62256 is a CMOS static RAM organized 32-kword × 8-bit. It realizes higher performance and low power consumption by employing 0.8 µm Hi-CMOS process technology. The device, packaged in 8 × 14 mm TSOP, 8 × 13.4 mm TSOP with thickness of 1.2 mm, 450-mil SOP (foot print pitch width), 600-mil plastic DIP, or 300-mil plastic DIP, is available for high density mounting. It offers low power standby power dissipation; therefore, it is suitable for battery back-up systems.

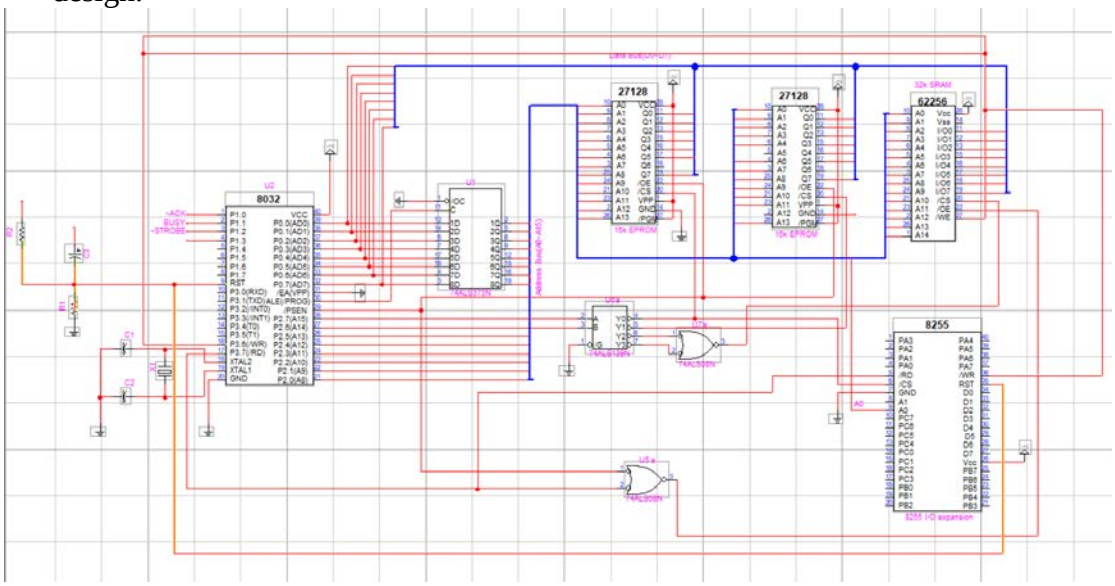
8255:

The 8255 is a general purpose programmable I/O device which may be used with many different microprocessors. There are 24 I/O pins which may be individually programmed in 2 groups of 12 and used in 3 major modes of operation.

(b) Look up and use necessary TTL ICs for interfacing those memory chips (e.g., latch for ALE, decoder for /CS for each memory chips and 8255, etc.).
Answer:
16KB EPROM: 27128
32KB SRAM: 62256

Latch for ALE: 74373
Decoder for /CS and 8255: 74139 -> By using the 74139 address decoder, we can simplify the logic with using less AND gates.

(c) Power supply, reset circuit, and crystal oscillator should also be included in your design.



You should try to read class materials (8051 hardware) and the data books, and select proper ICs. Proper documentation is required for obtaining high grades.