List #:

THE UNIVERSITY OF BRITISH COLUMBIA

Department of Electrical and Computer Engineering

CPEN312 Final Exam – April 11, 2018

Answer all problems.
Time: 2.5 Hours.
This examination consists of 11 pages. Please check that you have a complete copy. You may use both sides of each sheet if needed.

NOT Permitted: CELLPHONES, COMPUTERS, or other ELECTRONIC AID DEVICES.
Permitted: Calculators, Notes, and books.

Name:

Student Number:

#	MAX	GRADE
1	10	
2	10	
3	10	
4	15	
5	10	
6	10	
7	15	
8	20	
TOTAL	100	

READ THIS

IMPORTANT NOTE: The announcement "stop writing" will be made at the end of the examination. Anyone writing after this announcement will receive a score of 0. No exceptions, no excuses.

All writings must be on this booklet. The blank sides on the reverse of each page may also be used.

Each candidate should be prepared to produce, upon request, his/her Library/AMS card.

Read and observe the following rules:
No candidate shall be permitted to enter the examination room after the expiration of one-half hour, or to leave during the first half-hour of the examination.

Candidates are not permitted to ask questions of the invigilators, except in cases of supposed errors or ambiguities in examination-questions.

***Caution** - Candidates guilty of any of the following, or similar, dishonest practices shall be immediately dismissed from the examination and shall be liable to disciplinary action:*

- *Making use of any books, papers or memoranda, calculators, audio or visual cassette players or other memory aid devices, other than as authorized by the examiners.*
- *Speaking or communicating with other candidates.*
- *Purposely exposing written papers to the view of other candidates.*

The plea of accident or forgetfulness shall not be received.

1) Assemble by hand the following subroutine for the CV-8052 processor. Use the opcodes provided in the appendices at the end of this exam. (10 marks)

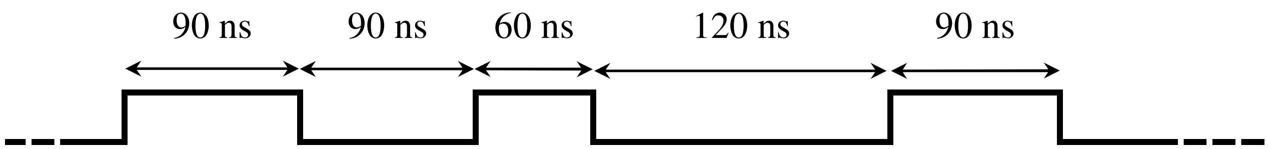
```
| Address | Opcode/Operands | Instructions
|---------+-----------------+-------------------------
| 0030    |                 |         dseg at 30H
| 0030    |                 | bcd:    ds 5
| 0035    |                 | x:      ds 4
| 1000    |                 |         cseg at 1000H
| 1000    | E4              | h2b:    clr a
| 1001    | F5 30           |         mov bcd+0, a
| 1003    | F5 31           |         mov bcd+1, a
| 1005    | F5 32           |         mov bcd+2, a
| 1007    | F5 33           |         mov bcd+3, a
| 1009    | F5 34           |         mov bcd+4, a
| 100B    | 7A 20           |         mov r2, #32
| 100D    | 79 04           | h2b_L0: mov r1, #4
| 100F    | 78 35           |         mov r0, #(x+0)
| 1011    | E6              | h2b_L1: mov a, @r0
| 1012    | 33              |         rlc a
| 1013    | F6              |         mov @r0, a
| 1014    | 08              |         inc r0
| 1015    | D9 FA           |         djnz r1, h2b_L1
| 1017    | 79 05           |         mov r1, #5
| 1019    | 78 30           |         mov r0, #(bcd+0)
| 101B    | E6              | h2b_L2: mov a, @r0
| 101C    | 36              |         addc a, @r0
| 101D    | D4              |         da a
| 101E    | F6              |         mov @r0, a
| 101F    | 08              |         inc r0
| 1020    | D9 F9           |         djnz r1, h2b_L2
| 1022    | DA E9           |         djnz r2, h2b_L0
| 1024    | 22              |         ret
```

2) Disassemble the following sequence of machine code for the 8051 microcontroller. All the numbers are in hexadecimal. Use the tables of opcodes provided in the appendices at the end of this exam. (10 marks)

C0 D0 C0 E0 C0 00 E9 29 24 32 F8 D8 FE D0 00 D0 E0 D0 D0 22

```
| Address | Opcode/Operands | Instruction
|---------+-----------------+------------------------
| 0000    |                 | cseg at 0000h
| 0000    | C0D0            |         push psw
| 0002    | C0E0            |         push acc
| 0004    | C000            |         push 0h
| 0006    | E9              |         mov a, r1
| 0007    | 29              |         add a, r1
| 0008    | 2432            |         add a, #32h
| 000A    | F8              |         mov r0, a
| 000B    |                 | L0001:
| 000B    | D8FE            |         djnz r0, L0001
| 000D    | D000            |         pop 0h
| 000F    | D0E0            |         pop acc
| 0011    | D0D0            |         pop psw
| 0013    | 22              |         ret
| 0014    |                 | end
```

3) Write a subroutine to generate the signal shown below at pin P0.0 of a CV-8052 processor running at 33.333333 MHz. The CV-8052 takes one clock period per machine cycle. Assume the pin is configured as an output, and that the signal is already set to logic zero before the subroutine is called. Use the cycles per instruction in the tables provided at the end of this exam. (10 marks)




```
; Each cycle is 30ns
gen_signal:
    setb P0.0 ; Pin goes high (First 2 cycles are before start of signal)
    nop       ; Wait 1 cycle
    clr P0.0  ; Pin goes low after 2 cycles
    nop       ; Wait 1 cycle
    setb P0.0 ; Pin goes high after 2 cycles
    clr P0.0  ; Pin goes low after 2 cycles
    nop       ; Wait 1 cycle
    nop       ; Wait 1 cycle
    setb P0.0 ; Pin goes high after 2 cycles
    nop       ; Wait 1 cycle
    clr P0.0  ; Pin goes low after 2 cycles
    ret       ; Return 3 cycles later
```

4) Write an assembly subroutine for the 8051 microcontroller to perform the operation Z=X−Y, where Z, X, and Y are packed **BCD** numbers defined as:

```
        DSEG at 040H
X:      DS 3 ; six BCD digits for input X
Y:      DS 3 ; six BCD digits for input Y
        XSEG at 4000H
Z:      DS 3 ; six BCD digits for result Z
```

Assume the least significant **BCD** digits are stored at the lowest memory location for all the variables. Tip: X−Y =X + nine_complement(Y) + 1. (15 marks)

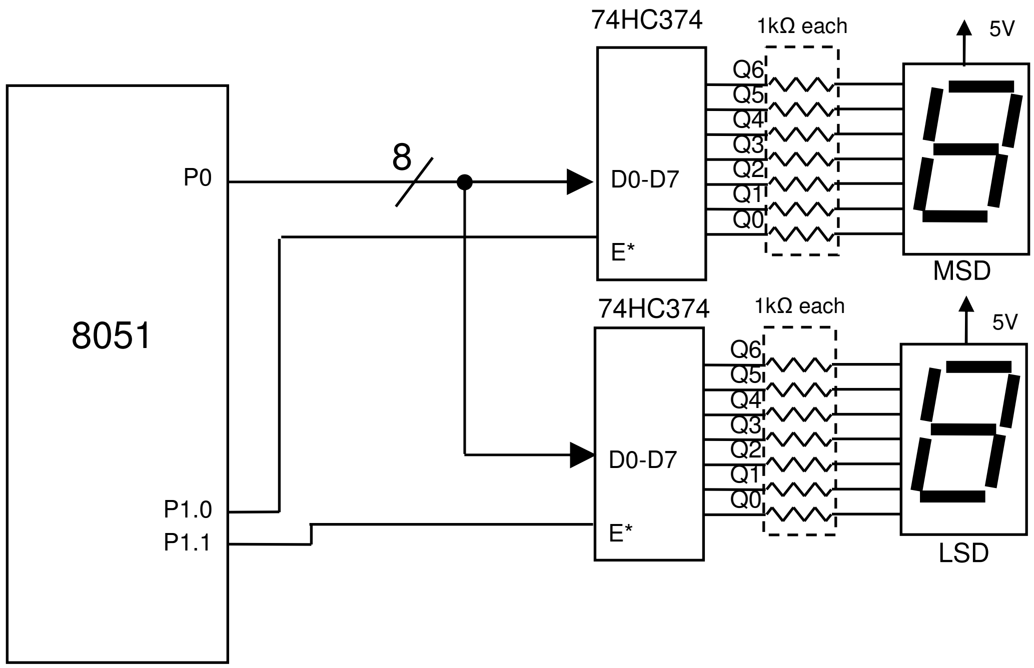
```
ninecpl MAC
    mov A, #99H
    clr C
    subb A, %0
    mov %0, A
ENDMAC

addbcd MAC
    mov A, %0
    addc A, %1
    da A
ENDMAC

movZ MAC
    mov dptr, %0
    movx @dptr, A
ENDMAC

six_dig_sub:
    ninecpl(Y+0)
    ninecpl(Y+1)
    ninecpl(Y+2)
    setb C
    addbcd(X+0, Y+0)
    movZ(#Z+0)
    addbcd(X+1, Y+1)
    movZ(#Z+1)
    addbcd(X+2, Y+2)
    movZ(#Z+2)
    ret
```

5) A little league baseball scoreboard includes the circuit shown in the figure below to count the number of pitches the current pitcher has thrown. Write an 8051 assembly subroutine to increment the number of pitches stored in register B (in BCD) and display the result using the common-anode 7-segment displays wired as shown in the circuit. (10 marks)




```
Dec7Seg:
    DB 40H, 79H, 24H, 30H, 19H, 12H, 02H, 78H, 00H, 10H
```

```
writeDec MAC
    ; Grab 7seg from LUT
    movc A, @A+dptr
    ; Write out to latch
    mov P0, A
    ; Enable latch (active rising edge)
    setb %0
ENDMAC
```

```
incBandDisp:
    ; Set CLK low
    clr P1.0
    clr P1.1
    ; Point to LUT
    mov dptr, #Dec7Seg
    ; BCD increment B
    mov A, B
    inc A
    da A
    mov B, A
    ; Mask lower nibble
    anl A, #1111b
    ; Write out to LSD display
    writeDec(P1.1)
    ; Mask higher nibble
    mov A, B
    swap A
    anl A, #1111b
    ; Write out to MSD display
    writeDec(P1.0)
    ret
```

6) The two look-up tables below can be used to quickly convert a register to its hexadecimal ASCII representation. Write an assembly subroutine for the 8051 microcontroller to convert the value passed in register B to its hexadecimal ASCII representation and store the two ASCII values into registers R6 and R7 where R6 is the least significant digit. If variable UPPER is set to '1' use upper case letters. Otherwise use lower case letters. (10 marks)

```
BSEG
UPPER: DBIT 1

CSEG
TO_HEX_UPPER: DB '0123456789ABCDEF'
TO_HEX_LOWER: DB '0123456789abcdef'
```

```
hex2ascii:
    mov dptr, #TO_HEX_UPPER
    jb UPPER, not_lower
    mov dptr, #TO_HEX_LOWER
not_lower:
    mov A, B
    anl A, #1111B
    movc A, @A+dptr
    mov R6, A
    mov A, B
    swap A
    anl A, #1111B
    movc A, @A+dptr
    mov R7, A
```

7) The 8051 assembly subroutine below configures timer/counter 0 as a 16-bit counter. Write an Interrupt Service Routine for counter 0 that:

a) Increments a 16-bit variable defined in the ISEG called ‘ovf_count’. Assume the variable is located in memory above address 0x7F.
b) Preserves the value of all the used registers.
c) Returns properly.

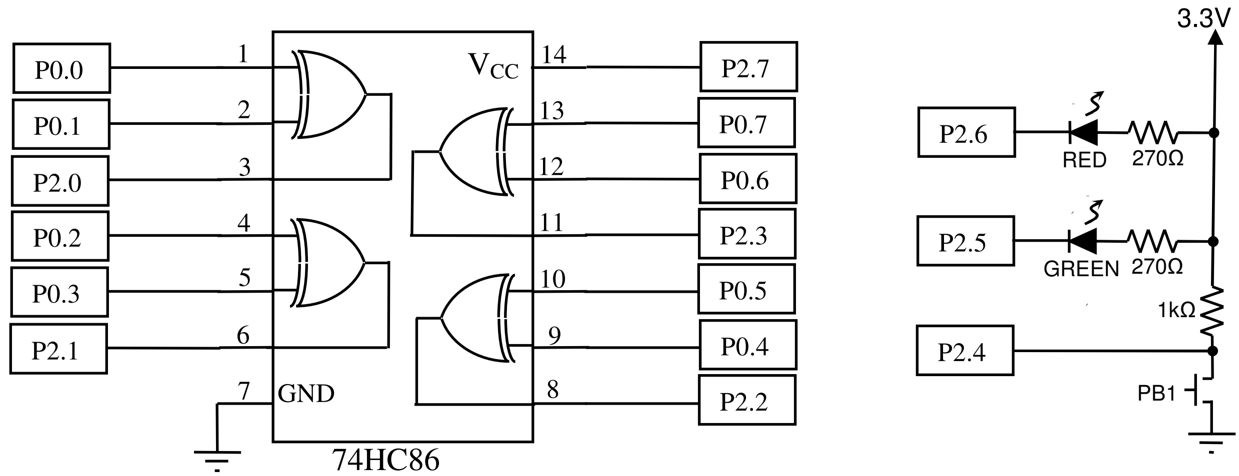
Note: the vector address for counter 0 interrupt is 000BH. (15 marks)

```
Init_counter_0:
    clr EA          ; Disable interrupts
    clr TR0         ; Stop counter 0
    mov TMOD, #05H ; Configure counter 0 in mode 1
    clr TF0         ; Clear overflow flag
    mov TH0, #0
    mov TL0, #0
    clr a
    mov R0, #ovf_count
    mov @R0, a      ; Clear overflow counter low
    inc R0
    mov @R0, a      ; Clear overflow counter high
    setb TR0        ; Start counter 0
    setb ET0        ; Enable counter 0 overflow interrupt
    setb EA         ; Enable global interrupts
    ret

org 000BH
    ljmp inc_ovf
    reti

inc_ovf:
    push ACC
    push PSW ; Carry flag is stored in PSW
    clr C
    mov A, ovf_count+0
    addc A, #1
    mov ovf_count+0, A
    mov A, ovf_count+1
    addc A, #0
    mov ovf_count+1, A
    pop PSW
    pop ACC
    ret
```

8) The circuit in the figure below can be used to verify that a 74HC86 integrated circuit (IC) operates correctly. The 74HC86 IC consists of four 2-input XOR gates. Write a subroutine for the CV-8052 processor that tests the IC after push button PB1 is pressed and then released. Additionally, the subroutine should: configure the input and output pins, apply power to the IC, test all possible input/output combinations and either turn the green LED on if the IC passes all the tests or the red LED on if the IC fails any test. The subroutine should set the power pin as well as all of the IC inputs to zero before returning so that the IC can be removed safely from the circuit after the tests are completed. TIP: you can test all four gates at once! (20 marks)




```
truth_table:
    db 0, 1, 1, 0

acc2xor:
    push ACC
    mov B, #0x55
    mul AB ; Fill accumulator with lower 2 bits
    mov P0, A ; Write outputs to P0
    pop ACC
    ret

cmpxor:
    push ACC
    movc A, @A+dptr ; Move answer into accumulator
    mov B, #0xF
    mul AB ; Fill lower 4 bits with answer
    orl A, #0xF0 ; Prevent underflow issues
    clr C
    subb A, P0
    clr C
    anl A, #0x0F ; Only care about lower 4 bits
    jz noerror
    setb C ; Set carry if input is different from truth table
noerror:
    pop ACC
    ret
```

```
check_xor:
    mov dptr, #truth_table
    mov P0MOD, #0xFF
    mov P2MOD, #0x1F
    orl P2MOD, #0x60 ; Disable LEDs
wait_press:
    jb P2.4, wait_press
wait_release:
    jnb P2.4, wait_release
    setb P2.7 ; Power chip
    mov A, #4
loop_zoop:
    dec A
    lcall acc2xor
    lcall cmpxor
    jc invalid ; Bad chip if carry is set
    jnz loop_zoop
valid:
    clr P2.5
    sjmp cleanup
invalid:
    clr P2.6
    sjmp cleanup
cleanup:
    clr P2.7
    mov P0, #0
```

Appendix 1: CV-8052 Instructions Sorted by Opcode Number

Opcode	Hex	C	B	Mnemonic
00000000	0x00	1	1	NOP
aaa00001		3	2	AJMP paged_addr
00000010	0x02	3	3	LJMP abs_addr
00000011	0x03	1	1	RR A
00000100	0x04	1	1	INC A
00000101	0x05	2	2	INC data
0000011i	0x06–0x07	1	1	INC @Ri
00001rrr	0x08–0x0F	1	1	INC Rn
00010000	0x10	3/4	3	JBC bit,rel
aaa10001		3	2	ACALL paged_addr
00010010	0x12	3	3	LCALL abs_addr
00010011	0x13	1	1	RRC A
00010100	0x14	1	1	DEC A
00010101	0x15	2	2	DEC data
0001011i	0x16–0x17	1	1	DEC @Ri
00011rrr	0x18–0x1F	1	1	DEC Rn
00100000	0x20	3/4	3	JB bit,rel
00100010	0x22	3	1	RET
00100011	0x23	1	1	RL A
00100100	0x24	2	2	ADD A,#val
00100101	0x25	2	2	ADD A,data
0010011i	0x26–0x27	1	1	ADD A,@Ri
00101rrr	0x28–0x2F	1	1	ADD A,Rn
00110000	0x30	3/4	3	JNB bit,rel
00110010	0x32	3	1	RETI
00110011	0x33	1	1	RLC A
00110100	0x34	2	2	ADDC A,#val
00110101	0x35	2	2	ADDC A,data
0011011i	0x36–0x37	1	1	ADDC A,@Ri
00111rrr	0x38–0x3F	1	1	ADDC A,Rn
01000000	0x40	2/3	2	JC rel
01000010	0x42	2	2	ORL data,A
01000011	0x43	3	3	ORL data,#val
01000100	0x44	2	2	ORL A,#val
01000101	0x45	2	2	ORL A,data
0100011i	0x46–0x47	1	1	ORL A,@Ri
01001rrr	0x48–0x4F	1	1	ORL A,Rn
01010000	0x50	2/3	2	JNC rel
01010010	0x52	2	2	ANL data,A
01010011	0x53	3	3	ANL data,#val
01010100	0x54	2	2	ANL A,#val
01010101	0x55	2	2	ANL A,data
0101011i	0x56–0x57	1	1	ANL A,@Ri
01011rrr	0x58–0x5F	1	1	ANL A,Rn
01100000	0x60	2/3	2	JZ rel
01100010	0x62	2	2	XRL data,A
01100011	0x63	3	3	XRL data,#val
01100100	0x64	2	2	XRL A,#val
01100101	0x65	2	2	XRL A,data
0110011i	0x66–0x67	1	1	XRL A,@Ri
01101rrr	0x68–0x6F	1	1	XRL A,Rn
01110000	0x70	2/3	2	JNZ rel
01110010	0x72	2	2	ORL C,bit
01110011	0x73	2	1	JMP @A+DPTR
01110100	0x74	2	2	MOV A,#val
01110101	0x75	3	3	MOV data,#val

Opcode	Hex	C	B	Mnemonic
0111011i	0x76–0x77	2	2	MOV @Ri,#val
01111rrr	0x78–0x7F	2	2	MOV Rn,#val
10000000	0x80	3	2	SJMP rel
10000010	0x82	2	2	ANL C,bit
10000011	0x83	4	1	MOVC A,@A+PC
10000100	0x84	10	1	DIV AB
10000101	0x85	3	3	MOV data,data
1000011i	0x86–0x87	2	2	MOV data,@Ri
10001rrr	0x88–0x8F	2	2	MOV data,Rn
10010000	0x90	3	3	MOV DPTR,#val
10010010	0x92	2	2	MOV bit,C
10010011	0x93	4	1	MOVC A,@A+DPTR
10010100	0x94	2	2	SUBB A,#val
10010101	0x95	2	2	SUBB A,data
1001011i	0x96–0x97	1	1	SUBB A,@Ri
10011rrr	0x98–0x9F	1	1	SUBB A,Rn
10100000	0xA0	2	2	ORL C,/bit
10100010	0xA2	2	2	MOV C,bit
10100011	0xA3	1	1	INC DPTR
10100100	0xA4	1	1	MUL AB
1010011i	0xA6–0xA7	3	2	MOV @Ri,data
10101rrr	0xA8–0xAF	3	2	MOV Rn,data
10110000	0xB0	2	2	ANL C,/bit
10110010	0xB2	2	2	CPL bit
10110011	0xB3	1	1	CPL C
10110100	0xB4	3/4	3	CJNE A,#val,rel
10110101	0xB5	3/4	3	CJNE A,data,rel
1011011i	0xB6–0xB7	3/4	3	CJNE @Ri,#val,rel
10111rrr	0xB8–0xBF	3/4	3	CJNE Rn,#val,rel
11000000	0xC0	3	2	PUSH data
11000010	0xC2	2	2	CLR bit
11000011	0xC3	1	1	CLR C
11000100	0xC4	1	1	SWAP A
11000101	0xC5	2	2	XCH A,data
1100011i	0xC6–0xC7	1	1	XCH A,@Ri
11001rrr	0xC8–0xCF	1	1	XCH A,Rn
11010000	0xD0	3	2	POP data
11010010	0xD2	2	2	SETB bit
11010011	0xD3	1	1	SETB C
11010100	0xD4	1	1	DA A
11010101	0xD5	3/4	3	DJNZ data,rel
1101011i	0xD6–0xD7	1	1	XCHD A,@Ri
11011rrr	0xD8–0xDF	2/3	2	DJNZ Rn,rel
11100000	0xE0	2	1	MOVX A,@DPTR
1110001i	0xE2–0xE3	2	1	MOVX A,@Ri
11100100	0xE4	1	1	CLR A
11100101	0xE5	2	2	MOV A,data
1110011i	0xE6–0xE7	1	1	MOV A,@Ri
11101rrr	0xE8–0xEF	1	1	MOV A,Rn
11110000	0xF0	1	1	MOVX @DPTR,A
1111001i	0xF2–0xF3	1	1	MOVX @Ri,A
11110100	0xF4	1	1	CPL A
11110101	0xF5	2	2	MOV data,A
1111011i	0xF6–0xF7	1	1	MOV @Ri,A
11111rrr	0xF8–0xFF	1	1	MOV Rn,A

Appendix 2: CV-8052 Instructions Sorted by Name

Opcode	Hex	C	B	Mnemonic
aaa10001		3	2	ACALL paged_addr
00100100	0x24	2	2	ADD A,#val
0010011i	0x26-0x27	1	1	ADD A,@Ri
00100101	0x25	2	2	ADD A,data
00101rrr	0x28-0x2F	1	1	ADD A,Rn
00110100	0x34	2	2	ADDC A,#val
0011011i	0x36-0x37	1	1	ADDC A,@Ri
00110101	0x35	2	2	ADDC A,data
00111rrr	0x38-0x3F	1	1	ADDC A,Rn
aaa00001		3	2	AJMP paged_addr
01010100	0x54	2	2	ANL A,#val
0101011i	0x56-0x57	1	1	ANL A,@Ri
01010101	0x55	2	2	ANL A,data
01011rrr	0x58-0x5F	1	1	ANL A,Rn
10110000	0xB0	2	2	ANL C,/bit
10000010	0x82	2	2	ANL C,bit
01010011	0x53	3	3	ANL data,#val
01010010	0x52	2	2	ANL data,A
1011011i	0xB6-0xB7	3/4	3	CJNE @Ri,#val,rel
10110100	0xB4	3/4	3	CJNE A,#val,rel
10110101	0xB5	3/4	3	CJNE A,data,rel
10111rrr	0xB8-0xBF	3/4	3	CJNE Rn,#val,rel
11100100	0xE4	1	1	CLR A
11000010	0xC2	2	2	CLR bit
11000011	0xC3	1	1	CLR C
11110100	0xF4	1	1	CPL A
10110010	0xB2	2	2	CPL bit
10110011	0xB3	1	1	CPL C
11010100	0xD4	1	1	DA A
0001011i	0x16-0x17	1	1	DEC @Ri
00010100	0x14	1	1	DEC A
00010101	0x15	2	2	DEC data
00011rrr	0x18-0x1F	1	1	DEC Rn
10000100	0x84	10	1	DIV AB
11010101	0xD5	3/4	3	DJNZ data,rel
11011rrr	0xD8-0xDF	2/3	2	DJNZ Rn,rel
0000011i	0x06-0x07	1	1	INC @Ri
00000100	0x04	1	1	INC A
00000101	0x05	2	2	INC data
10100011	0xA3	1	1	INC DPTR
00001rrr	0x08-0x0F	1	1	INC Rn
00100000	0x20	3/4	3	JB bit,rel
00010000	0x10	3/4	3	JBC bit,rel
01000000	0x40	2/3	2	JC rel
01110011	0x73	2	1	JMP @A+DPTR
00110000	0x30	3/4	3	JNB bit,rel
01010000	0x50	2/3	2	JNC rel
01110000	0x70	2/3	2	JNZ rel
01100000	0x60	2/3	2	JZ rel
00010010	0x12	3	3	LCALL abs_addr
00000010	0x02	3	3	LJMP abs_addr
0111011i	0x76-0x77	2	2	MOV @Ri,#val
1111011i	0xF6-0xF7	1	1	MOV @Ri,A
1010011i	0xA6-0xA7	3	2	MOV @Ri,data
01110100	0x74	2	2	MOV A,#val
1110011i	0xE6-0xE7	1	1	MOV A,@Ri

Opcode	Hex	C	B	Mnemonic
11100101	0xE5	2	2	MOV A,data
11101rrr	0xE8-0xEF	1	1	MOV A,Rn
10010010	0x92	2	2	MOV bit,C
10100010	0xA2	2	2	MOV C,bit
01110101	0x75	3	3	MOV data,#val
1000011i	0x86-0x87	2	2	MOV data,@Ri
11110101	0xF5	2	2	MOV data,A
10000101	0x85	3	3	MOV data,data
10001rrr	0x88-0x8F	2	2	MOV data,Rn
10010000	0x90	3	3	MOV DPTR,#val
01111rrr	0x78-0x7F	2	2	MOV Rn,#val
11111rrr	0xF8-0xFF	1	1	MOV Rn,A
10101rrr	0xA8-0xAF	3	2	MOV Rn,data
10010011	0x93	4	1	MOVC A,@A+DPTR
10000011	0x83	4	1	MOVC A,@A+PC
11110000	0xF0	1	1	MOVX @DPTR,A
1111001i	0xF2-0xF3	1	1	MOVX @Ri,A
11100000	0xE0	2	1	MOVX A,@DPTR
1110001i	0xE2-0xE3	2	1	MOVX A,@Ri
10100100	0xA4	1	1	MUL AB
00000000	0x00	1	1	NOP
01000100	0x44	2	2	ORL A,#val
0100011i	0x46-0x47	1	1	ORL A,@Ri
01000101	0x45	2	2	ORL A,data
01001rrr	0x48-0x4F	1	1	ORL A,Rn
10100000	0xA0	2	2	ORL C,/bit
01110010	0x72	2	2	ORL C,bit
01000011	0x43	3	3	ORL data,#val
01000010	0x42	2	2	ORL data,A
11010000	0xD0	3	2	POP data
11000000	0xC0	3	2	PUSH data
00100010	0x22	3	1	RET
00110010	0x32	3	1	RETI
00100011	0x23	1	1	RL A
00110011	0x33	1	1	RLC A
00000011	0x03	1	1	RR A
00010011	0x13	1	1	RRC A
11010010	0xD2	2	2	SETB bit
11010011	0xD3	1	1	SETB C
10000000	0x80	3	2	SJMP rel
10010100	0x94	2	2	SUBB A,#val
1001011i	0x96-0x97	1	1	SUBB A,@Ri
10010101	0x95	2	2	SUBB A,data
10011rrr	0x98-0x9F	1	1	SUBB A,Rn
11000100	0xC4	1	1	SWAP A
1100011i	0xC6-0xC7	1	1	XCH A,@Ri
11000101	0xC5	2	2	XCH A,data
11001rrr	0xC8-0xCF	1	1	XCH A,Rn
1101011i	0xD6-0xD7	1	1	XCHD A,@Ri
01100100	0x64	2	2	XRL A,#val
0110011i	0x66-0x67	1	1	XRL A,@Ri
01100101	0x65	2	2	XRL A,data
01101rrr	0x68-0x6F	1	1	XRL A,Rn
01100011	0x63	3	3	XRL data,#val
01100010	0x62	2	2	XRL data,A