THE UNIVERSITY OF BRITISH COLUMBIA

Department of Electrical and Computer Engineering
CPEN312 Practice Final Exam

Answer all problems.
Time: 2.5 Hours.
This examination consists of 13 pages. Please check that you have a complete copy. You may use both sides of each sheet if needed.

NOT Permitted: CALCULATORS, CELLPHONES, ELECTRONIC AIDS.
Permitted: Books and notes.

Name:

Student Number:

#	MAX	GRADE
1	10	
2	10	
3	10	
4	10	
5	10	
6	10	
7	10	
8	10	
9	10	
10	10	
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 program for the CV-8052 processor. Use the opcodes provided in the appendices at the end of this exam. (10 marks)

Address	Opcode/Operands	Instruction
3000H	—	`org 3000H`
3000H	—	`BCD_X_20:`
3000H	—	`; BCD*2`
3000H	EC H	`MOV A, R4`
3001H	2C H	`ADD A, R4`
3002H	D4 H	`DA A`
3003H	FC H	`MOV R4, A`
3004H	ED H	`MOV A, R5`
3005H	3D H	`ADDC A, R5`
3006H	D4 H	`DA A`
3007H	FD H	`MOV R5, A`
3008H	—	`; Multiply BCD*2 by 10`
3009H	79 H, 04H	`MOV R1, #4`
300AH	C3 H	`L1: CLR C`
300BH	EC H	`MOV A, R4`
300CH	33 H	`RLC A`
300DH	FC H	`MOV R4, A`
300EH	ED H	`MOV A, R5`
300FH	33 H	`RLC A`
3010H	FD H	`MOV R5, A`
3011H	D9 H, F7H	`DJNZ R1, L1`
3013H	22 H	`RET`

next 3013H

des. 300AH

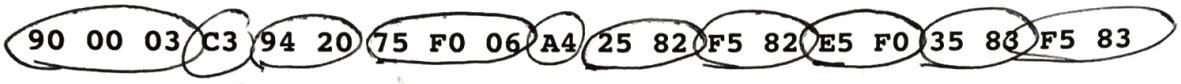
next + x = desired

next = 0011 0000 0001 0011

des. = 0011 0000 0000 1010

x = 1111 1111 1111 0111

2) Knowing that the SFRs B, DPL, and DPH are respectively located at addresses F0H, 82H, and 83H, disassemble the following sequence of machine language 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)



MOV DPTR, #03H
C/z C
SUBB A, #20H
MOV B, #06H
MUL AB
ADD A, DPL
MOV DPL, A
MOV A, B
ADDC A, DPH
MOV DPH, A

3) The subroutine below runs in a CV-8052 processor at 33.33MHz, which takes one clock per cycle (therefore, one cycle takes 30 ns). It is also known that $0 < R1 < 100$. Answer the questions that follow. You may use the tables of opcodes at the end of this exam.

```
Wait:
     push psw   3
     push acc   3
     push AR0   3
     mov a, R1  1
     add a, R1  1
     add a, #50 2
     mov R0, a  1
W1:  djnz R0, W1  2×129 +2
     pop AR0    3
     pop acc    3
     pop psw    3
     ret        3
```

$T = 30ns \times (136 \times 3 + 7) =$

$= 30 \times 415 = 12450\ ns =$

$= 12.45\mu s$

a) If the subroutine is called using the two assembly instructions below, how much time does the subroutine take to run? (7 marks)

```
mov R1, #40   2
lcall Wait    3
```

b) Write the equation that describes the run time of the 'Wait' subroutine above as a function of register R1. (3 marks)

$T = 30 \times ((2R_1 + 50 - 1) \times 3 + 28) = 30 \times (6R_1 + 147 + 28) =$

$= 180R_1 + 175 \times 30 = 180R_1 + 5250$

4) Write a SHORT (fewer bytes as possible) assembly subroutine for the 8051 microcontroller to perform the operation R=M-S, where R, M, and S are defined as:

```
        DSEG at 40H
M:      DS 8
S:      DS 8
R:      DS 8
```

Assume the least significant bit is stored at the lowest memory location for all the variables. NOTE: Marks will not be given for brute force solutions (10 marks)

M: (40, 41, 42, 43, 44, 45, 46, 47) H

S: (48, 49, 4A, 4B, 4C, 4D, 4E, 4F) H

R: (50, 51, 52, 53, 54, 55, 56, 57) H

Subtract:

```
        MOV  R0, #8
        clr C
        clr a
        MOV R1, #40H
        MOV R2, #48H
        MOV R3, #50H
L0:     MOV a,@R1
        SUBB a,@R2
        MOV @R3,a
        INC R1
        INC R2
        INC R3
        djnz R0, L0
```

```
        clr  C
        MOV  R0, #8 H
        MOV  R1, #40H
        MOV  R2, #48H
        MOV  R3, #50H
L1:     MOV A,@R1
        SUBB A,@R2
        MOV @R3,A
        INC R1
        INC R2
        INC R3
        djnz R0, L1
```

5) Write an assembly subroutine for the 8051 microcontroller to compute the average of eight consecutive 16-bit numbers stored from memory locations 40H to 4FH. Store the average in registers DPL (LSD) and DPH (MSD). Tip: use the 'rrc' instruction to divide a multi-byte number by a power of 2. (10 marks)

6) The look-up table 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 it into registers R6 and R7 where R6 is the least significant digit. (10 marks)

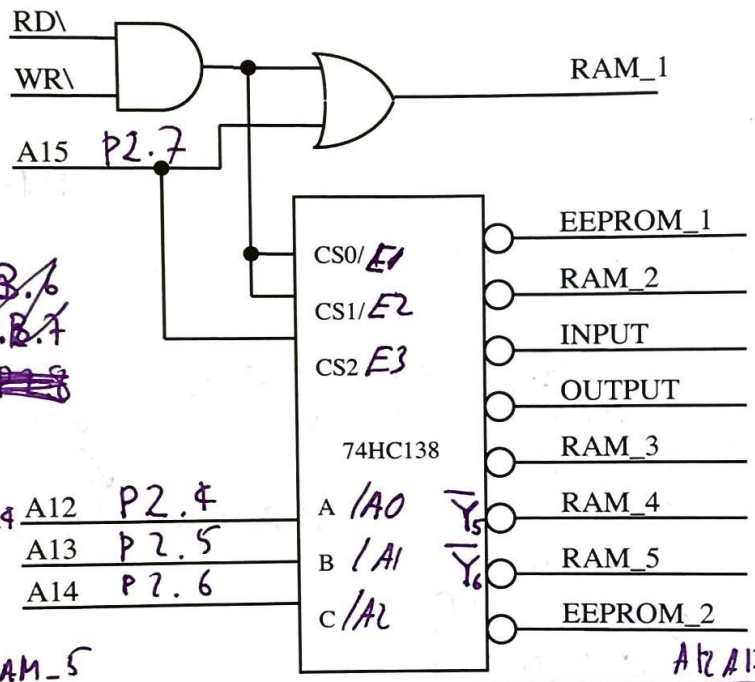
```
CSEG

TO_HEX: DB  '0', '1', '2', '3', '4', '5', '6', '7'
        DB  '8', '9', 'A', 'B', 'C', 'D', 'E', 'F'
```

Convert :

```
mov   DPTR, #TO_HEX
mov   A, B
swap  A          ; high part first
anl   a, #0FH    ; and w/ 00001111
movc  a, @A+DPTR
mov   R7, A      ; R7 : Hex('3')
mov   A, B
anl   A, #0FH
movc  A, @A+DPTR
mov   R6, A      ; Hex('F')
sjmp  $          ; loop forever
```

R7 : Hex('3')
R6 : Hex('F')

7) Consider the extended memory decoder for the 8051 microcontroller shown in the figure below where all the memory/devices chip enable signals are active low. Write an assembly subroutine to copy the first 75 bytes of memory 'RAM_4' to 'RAM_5'. Note: you must use the MOVX instruction. (10 marks)



#	E1	E2	E3	A0	A1	A2	Y5	Y6
RAM4	L	L	H	H	L	H	L	H
RAM5	L	L	H	L	H	H	H	L

```
Data_Transfer:  clr P3.7
                clr P3.6
                setb P2.7
                mov R0, #75
                mov DPTR, #RAM_4
                mov R1, DPL
                mov R2, DPH
                mov DPTR, #RAM_5
                mov R3, DPL
                mov R4, DPH
Mov_Ram_Loop:   setb P2.4
                setb P2.6
                clr P2.5
                mov DPL, R1
                mov DPH, R2
                movx A, @DPTR
                mov R5, A
```

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

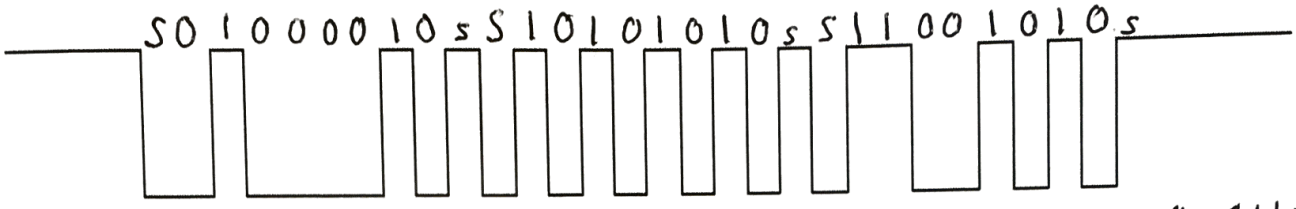
a) Reloads the timer to the same interrupt rate used in the initialization.
b) Adds the value read from port 0 to the value read from port 1 and writes the result to port 3. Assume the code will be running in a CV-8052 processor.
c) Preserves the value of all the used registers.
d) Returns properly.

Note: the vector address for timer 0 interrupt is 000BH. (10 marks)

```
Init_timer_0:
    clr EA; Disable interrupts
    mov TMOD, #01H; Configure timer 0 in mode 1
    clr TF0 ; Clear overflow flag
    clr TR0
    mov TH0, #high(1000H) → interrupt rate high
    mov TL0, #low(1000H)  → low
    setb TR0 ; Start timer 0
    mov P0MOD, #0 ; P0 is input → 0000 0000 B
    mov P1MOD, #0 ; P1 is input → 0000 0000 B
    mov P3MOD, #0FFH ; P3 is output → 1111 1111 B
    setb ET0 ; Enable timer 0 interrupt
    setb EA  ; Enable interrupts.
    ret
```

```
mov SP, #7FH
org 000BH
    ljmp add
add:
    push ACC
    push P0
    push P1
    mov TH0, #high(1000H)
    mov TL0, #low(1000H)
    mov A, P1
    add A, P0
    mov P3, A
    pop P1
    pop P0
    pop A
    reti
```

9) The asynchronous serial pattern in the figure shown below is received by a CV-8052 using the code that follows the figure. Determine the content of the 'rx' buffer after reaching the infinite loop. Tip: The ASCII for letter 'A' is 41H. (10 marks)

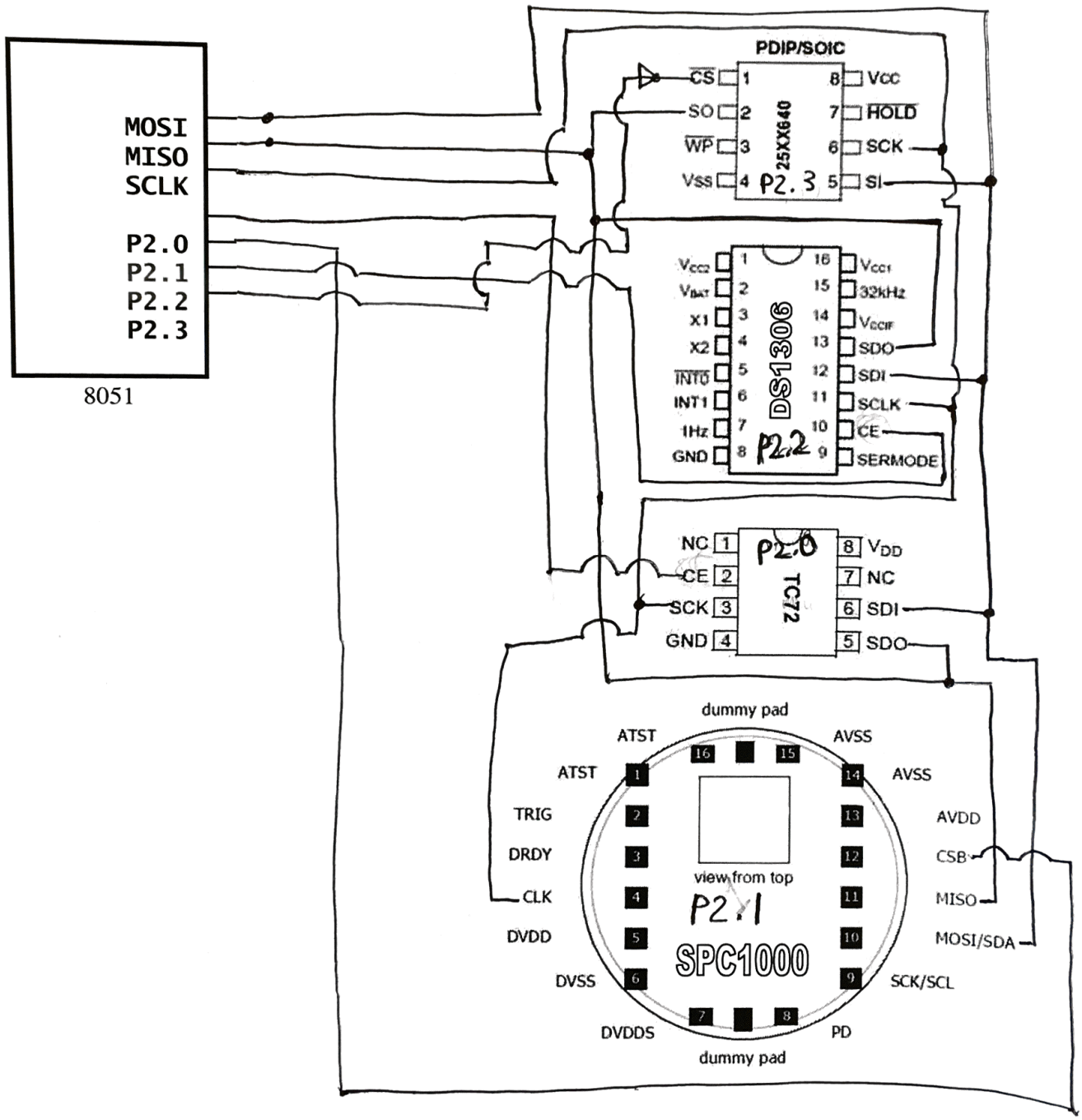



```
$MODDE0CV
org 0000H
    ljmp mp

DSEG at 30H
rx:  ds 3

FREQ    EQU 33333333
BAUD    EQU 115200
T2LOAD EQU 65536-(FREQ/(32*BAUD))

CSEG
isp: clr TR2
     mov T2CON, #30H
     mov RCAP2H, #high(T2LOAD)
     mov RCAP2L, #low(T2LOAD)
     setb TR2
     mov SCON, #52H
     ret

get: jnb RI, $
     clr RI
     mov @r0, SBUF
     inc r0
     ret

mp:  mov SP, #7FH
     mov r0, #rx
     lcall isp
     lcall get
     lcall get
     lcall get
     sjmp $
END
```

0100 0010 B = 42H : B
0101 0101 B = 55H : U
0101 0011 B = 53H : S

A 41H
B 42
C 43
D 44
E 45
F 46
G 47
H 48
I 49
J 4A
K 4B
L 4C
M 4D
N 4E
O 4F
P 50
Q 51
R 52
S 53
T 54
U 55

10) A weather station data logger is designed using an 8051 microcontroller and these four SPI peripherals: a TC32 temperature sensor, a SCP1000 barometric pressure sensor, a DS1306 real time clock, and a 25LC640 8k x 8 EEPROM. Show how to attach the peripherals to the microcontroller using the SPI bus. Do not worry about other signals present in the peripherals! Use P2.0, P2.1, P2.2, and P2.3 as slave chip enables for the TC32, SCP1000, DS1306, and 25LC640 respectively.



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