8051-실전 어셈블리 언어의 기초 이해와 문법

8051 어셈블리 언어

8051 어셈 명령어는 크게 5 종류로 나눌 수 있다.

(1). 산술 연산 명령어(Arithmetic Operation)

(2). 논리 연산 명령어(Logical Operation)

(3). 데이터 전송 명령어(Data Transfer Operation)

(4). 부울 조작 명령어(Boolean Manipulation Operation)

(5). 분기 명령어(Program Branching Operation)

- 산술 연산 명령어(Arithmetic Operation)
  - 증가/감소 명령(INC, DEC)
  - 덧셈/뺄셈 명령(ADD, ADDC, SUBB)
  - 10 진 조정 명령(DA A)
  - 곱셈/나눗셈 명령(MUL, DIV)
- 논리 연산 명령어(Logical Operation)
  - ANL(AND), ORL(OR), XRL(XOR), CLR(Clear), CPL(Complement)
  - RL, RLC, RR, RRC (Rotate), SWAP(Swap)
- 데이터 전송 명령어(Data Transfer Operation)
  - 내부 메모리 전송 명령 : MOV, PUSH, POP, XCH, XCHD
  - 외부 프로그램 전송 명령: MOVC
  - 외부 데이터 전송 명령: MOVX
- 부울 조작 명령어(Boolean Manipulation Operation)
  - 1 비트 데이터에 대한 논리 연산, 데이터 전송, SET/CLEAR 동작 명령
- 분기 명령어(Program Branching Operation)
  - 무조건, 조건, 반복 루프, 비교 브랜치 명령

8051 어셈블러 지시어(Directive)

'어셈블러 지시어'란 프로그램 실행과는 관계가 없고 단지 어셈블러에게 정보만 제공해 주는 명령어이다

1) ORG (Origin) : 프로그램과 데이터의 시작 번지 설정

2) END: 어셈블러 작업의 종료를 나타냄. END 문 이후의 어셈블리어 프로그램은 기계어로 변환되지 않는다

3) EQU (Equate) : Label 에 식의 값을 할당한다

4) DB (Define Byte) : 메모리에 데이터를 1 바이트 단위로 저장

5) DW (Define Word) : 메모리에 데이터를 2 바이트(워드) 단위로 저장

6) DS (Define Storage) : 메모리를 바이트 단위로 확보

Arithmetic Operations (산술 연산 명령어)

명령어	오퍼랜드	설 명	바이트	사이클
ADD	A, Rn	Acc 에 레지스터 Rn 의 값을 더함	1	1
	A, direct	Acc 에 direct 번지의 값을 더함	2	1
	A, @Ri	Acc 에 Ri 가 가리키는 번지의 값을 더함	1	1
	A, #data	Acc 에 data(상수) 값을 더함	2	1
ADDC	A, Rn	Acc 에 레지스터 Rn 의 값과 자리 올림 수를 함께 더함	1	1
	A, direct	Acc 에 direct 번지의 값과 자리올림수를 함께 더함	2	1
	A, @Ri	Acc 에 Ri 가 가리키는 번지의 값과 자리 올림수를 함께 더함	1	1
	A, #data	Acc 에 data(상수) 값과 자리 올림수를 함께 더함	2	1
SUBB	A, Rn	Acc 에서 레지스터 Rn 의 값과 빌림수를 함께 뺌	1	1
	A, direct	Acc 에서 direct 번지의 값과 빌림수를 함께 뺌	2	1
	A, @Ri	Acc 에서 Ri 가 가리키는 번지의 값과 빌림수를 함께 뺌	1	1
	A, #data	Acc 에서 data(상수) 값과 빌림수를 함께 뺌	2	1
INC	A	Acc 의 값을 1 증가	1	1
	Rn	레지스터 Rn 의 값을 1 증가	1	1
	direct	direct 번지의 값을 1 증가	2	1
	@Ri	Ri 가 가리키는 번지의 값을 1 증가	1	1
DEC	A	Acc 의 값을 1 감소	1	1
	Rn	레지스터 Rn 의 값을 1 감소	1	1
	direct	direct 번지의 값을 1 감소	2	1
	@Ri	Ri 가 가리키는 번지의 값을 1 감소	1	2
MUL	AB	A, B 레지스터를 곱하여 B 에 상위, A 에 하위 8 비트 저장	1	4
DIV	AB	A⁒B 를하여 A 레지스터에 몫, B 레지스터에 나머지 저장	1	4
DA	A	Acc 의 값을 BCD 코드 형태로 변환	1	1

Logical Operations (논리 연산 명령어)

명령어	오퍼랜드	설 명	바이트	사이클
ANL	A, Rn	Acc 와 레지스터 Rn 의 값을 AND 연산	1	1
	A, direct	Acc 와 direct 번지의 값을 AND 연산	2	1
	A, @Ri	Acc 와 Ri 가 가리키는 번지의 값을 AND 연산	1	1
	A, #data	Acc 와 data(상수) 값을 AND 연산	2	1
	direct, A	direct 번지의 값과 Acc 를 AND 연산	2	1
	direct, #data	direct 번지의 값과 data(상수) 값을 AND 연산	3	2
ORL	A, Rn	Acc 와 레지스터 Rn 의 값을 OR 연산	1	1
	A, direct	Acc 와 direct 번지의 값을 OR 연산	2	1
	A, @Ri	Acc 와 Ri 가 가리키는 번지의 값을 OR 연산	1	1
	A, #data	Acc 와 data(상수) 값을 OR 연산	2	1
	direct, A	direct 번지의 값과 Acc 를 OR 연산	2	1
	direct, #data	direct 번지의 값과 data(상수) 값을 OR 연산	3	2
XRL	A, Rn	Acc 와 레지스터 Rn 의 값을 EX-OR 연산	1	1
	A, direct	Acc 와 direct 번지의 값을 EX-OR 연산	2	1
	A, @Ri	Acc 와 Ri 가 가리키는 번지의 값을 EX-OR 연산	1	1
	A, #data	Acc 와 data(상수) 값을 EX-OR 연산	2	1
	direct, A	direct 번지의 값과 Acc 를 EX-OR 연산	2	1
	direct, #data	direct 번지의 값과 data(상수) 값을 EX-OR 연산	3	2
CLR	A	Acc 를 클리어 시킴	1	1
CPL	A	Acc 의 값을 컴플리먼트(1 의 보수) 시킴	1	1
RL	A	Acc 의 값을 왼쪽으로 로테이트(회전)	1	1

		시킴		
RLC	A	Acc 의 값을 캐리와 함께 왼쪽으로 로테이트(회전) 시킴	1	1
RR	A	Acc 의 값을 오른쪽으로 로테이트(회전) 시킴	1	1
RRC	A	Acc 의 값을 캐리와 함께 오른쪽으로 로테이트(회전) 시킴	1	1
SWAP	A	Acc 의 값의 상·하위 값을 교체시킴	1	1

Data Transfer (데이터 이동 명령어)

명령어	오퍼랜드	설 명	바이트	사이클
MOV	A, Rn	레지스터 Rn 의 값을 Acc 에 이동시킴	1	1
	A, direct	direct 번지의 값을 Acc 에 이동시킴	2	1
	A, @Ri	Ri 가 가리키는 번지의 값을 Acc 에 이동시킴	1	1
	Rn, #data	data(상수)의 값을 레지스터 Rn 에 이동시킴	2	1
	Rn, A	Acc 의 값을 레지스터 Rn 에 이동시킴	1	1
	Rn, direct	direct 번지의 값을 레지스터 Rn 에 이동시킴	2	2
	Rn, #data	data(상수)의 값을 레지스터 Rn 에 이동시킴	2	1
	direct, A	Acc 의 값을 direct 번지로 이동시킴	2	1
	direct, Rn	레지스터 Rn 의 값을 direct 번지로 이동시킴	2	2
	direct, direct	direct 번지의 값을 direct 번지로 이동시킴	3	2
	direct, @Ri	Ri 가 가리키는 번지의 값을 direct 번지로 이동시킴	2	2

	direct, #data	data(상수)의 값을 direct 번지로 이동시킴	3	2
	@Ri, A	Acc 의 값을 Ri 가 가리키는 번지로 이동시킴	1	1
	@Ri, direct	direct 번지의 값을 Ri 가 가리키는 번지로 이동시킴	2	2
	@Ri, #data	data(상수)의 값을 Ri 가 가리키는 번지로 이동시킴	2	1
	DPTR, #data16	16bit data(상수)의 값을 DPTR 로 이동시킴	3	2
MOVC	A, @A+DPTR	Acc 에 Acc+DPTR 이 가리키는 번지의 값을 이동시킴	1	2
	A, @A+PC	Acc 에 Acc+PC 가 가리키는 번지의 값을 이동시킴	1	2
MOVX	A, @Ri	Acc 에 Ri 가 가리키는 외부 번지의 값을 이동시킴	1	2
	A, @DPTR	Acc 에 DPTR 이 가리키는 외부 번지의 값을 이동시킴	1	2
	@Ri, A	Ri 가 가리키는 외부 번지에 Acc 의 값을 이동시킴	1	2
	@DPTR, A	DPTR 이 가리키는 외부 번지에 Acc 의 값을 이동시킴	1	2
PUSH	direct	direct 번지의 값을 스택에 넣음	2	2
POP	direct	스택의 SP 가 지시하는 위치의 값을 꺼내 direct 번지에 넣음	2	2
XCH	A, Rn	Acc 의 값과 레지스터 Rn 의 값을 교환	1	1
	A, direct	Acc 의 값과 direct 번지의 값을 교환	2	1
	A, @Ri	Acc 의 값과 Ri 가 가리키는 번지의 값을 교환	1	1

XCHD	A, @Ri	Acc 의 값과 Ri 가 가리키는 번지의 값의 하위 4bit 교환	1	1

Boolean Manipulation (불 대수 처리 명령어)

명령어	오퍼랜드	설 명	바이트	사이클
CLR	C	캐리 플래그를 클리어(0) 시킴	1	1
	bit	bit 어드레스의 값을 클리어(0) 시킴	2	1
SETB	C	캐리 플래그를 셋(1) 시킴	1	1
	bit	bit 어드레스의 값을 셋(1) 시킴	2	1
CPL	C	캐리 플래그를 컴플리먼트(1 의 보수) 시킴	1	1
	bit	bit 어드레스의 값을 컴플리먼트(1 의 보수) 시킴	2	1
ANL	C, bit	캐리 플래그와 bit 어드레스의 값을 AND 시킴	2	2
	C, /bit	캐리 플래그와 bit 어드레스의 컴플리먼트 값을 AND 시킴	2	2
ORL	C, bit	캐리 플래그와 bit 어드레스의 값을 OR 시킴	2	2
	C, /bit	캐리 플래그와 bit 어드레스의 컴플리먼트 값을 OR 시킴	2	2
MOV	C, bit	캐리 플래그에 bit 어드레스의 값을 이동시킴	2	1
	bit, C	bit 어드레스에 캐리 플래그의 값을 이동시킴	2	2

Program Branching Operation (서브루틴 / 분기 명령어)

명령어	오퍼랜드	설 명	바이트	사이클
ACALL	addr11	2Kbyte (2048 byte) 내의 절대 콜 (서브루틴)	2	2
LCALL	addr16	롱 콜 (서브루틴)	3	2
RET		서브루틴의 복귀	1	2
RET1		인터럽트로부터의 복귀	1	2
AJMP	addr11	2Kbyte (2048 byte) 내의 절대 점프	2	2
LJMP	addr16	롱 점프	3	2

SJMP	rel	현재 PC 에 변위(rel : -128~128)를 더한 위치로 점프	2	2
JMP	@A+DPTR	Acc 와 DPTR 의 값을 더한 주소로 점프	1	2
JZ	rel	Acc 가 0 이면 현재 PC 에 변위(rel : -128~128)를 더한 위치로 점프	2	2
JNZ	rel	Acc 가 0 이 아니면 PC 에 변위(rel : -128~128)를 더한 위치로 점프	2	2
JC	rel	캐리 플래그가 1 이면 PC 에 변위(rel : -128~128)를 더한 위치로 점프	2	2
JNC	rel	캐리 플래그가 0 이면 PC 에 변위(rel : -128~128)를 더한 위치로 점프	2	2
JB	bit, rel	비트 어드레스가 셋(1) 상태이면 PC 에 변위를 더한 위치로 점프	3	2
JNB	bit, rel	비트 어드레스가 셋(1) 상태가 아니면 PC 에 변위를 더한 위치로 점프	3	2
JBC	bit, rel	비트 어드레스가 셋(1) 상태이면 PC 에 변위를 더한 위치로 점프 & 비트 클리어(0)	3	2
CJNE	A, direct, rel	Acc 와 direct 번지의 값을 비교하여 다르면 PC 에 변위를 더한 위치로 점프	3	2
	A, #data, rel	Acc 와 data(상수) 값을 비교하여 다르면 PC 에 변위를 더한 위치로 점프	3	2
	Rn, #data, rel	레지스터 Rn 과 data(상수) 값을 비교하여 다르면 PC 에 변위를 더한 위치로 점프	3	2
	@Ri, #data, rel	Ri 가 가리키는 값과 data(상수) 값을 비교하여 다르면 PC 에 변위를 더한 위치로 점프	3	2
DJNZ	Rn, rel	레지스터 Rn 값을 1 감소시키고 PC 에 변위를 더한 위치로 점프	2	2
	direct, rel	direct 번지의 값을 1 감소시키고 PC 에 변위를 더한 위치로 점프	3	2
NOP		아무 처리도 하지 않음	1	1

오퍼랜드의 어드레스 모드 설명	
Rn	메모리 어드레스 00~1FH / 범용 레지스터
direct	메모리 어드레스 20H~2FH / 내부 데이터 메모리 (16byte = 128bit) /
@Ri	내부 데이터 메모리의 간접 번지 지정 / R0, R1 이용
#data	8bit 상수 데이터
#data16	16bit 상수 데이터
bit	비트 어드레스 영역의 비트 번호
addr16	LCALL 과 LJMP 에서 사용하는 64Kbyte 내의 프로그램 메모리 어드레스
addr11	ACALL 과 AJMP 에서 사용하는 2Kbyte 내의 프로그램 메모리 어드레스
rel	SJMP 등에서 사용하는 -128~128byte 사이의 어드레스 변위

다음의 어셈블 리스트는 C51 에 있는 A8051INS.S03 을 A8051.EXE 로 어셈한 리스트 파일을 보이고 있다. 필요시 참조 바람.

```
################################################################################
#                                                                              #
```

```
#      Micro Series 8051 Assembler V2.02/DOS                24/Nov/99    21:15:24          #
#                                                                                          #
#         Source     =     a8051ins.s03                                                    #
#         List       =     a.lst                                                           #
#         Object     =     a8051ins.r03                                                    #
#         Options    =                                                                     #
#                                                                                          #
#                                                          (c) Copyright IAR Systems 1990   #
##############################################################################
    1  0000                 aseg
    2  0000                 org     0
    3
    4  0000 00              nop
    5  0001 0123            ajmp    padr
    6  0003 028765          ljmp    ladr
    7  0006 03              rr      a
    8  0007 04              inc     a
    9  0008 0535            inc     dadr
   10  000A 06              inc     @r0
   11  000B 07              inc     @r1
   12  000C 08              inc     r0
   13  000D 09              inc     r1
   14  000E 0A              inc     r2
   15  000F 0B              inc     r3
   16  0010 0C              inc     r4
   17  0011 0D              inc     r5
   18  0012 0E              inc     r6
   19  0013 0F              inc     r7
   20  0014 1027F3          jbc     badr,$-10
   21  0017 1123            acall   padr
   22  0019 128765          lcall   ladr
   23  001C 13              rrc     a
   24  001D 14              dec     a
   25  001E 1535            dec     dadr
   26  0020 16              dec     @r0
   27  0021 17              dec     @r1
   28  0022 18              dec     r0
   29  0023 19              dec     r1
   30  0024 1A              dec     r2
   31  0025 1B              dec     r3
   32  0026 1C              dec     r4
   33  0027 1D              dec     r5
   34  0028 1E              dec     r6
   35  0029 1F              dec     r7
   36  002A 2027F3          jb      badr,$-10
   37  002D 21              db      21h             ;ajmp
   38  002E 22              ret
   39  002F 23              rl      a
   40  0030 2409            add     a,#imm8
   41  0032 2535            add     a,dadr
   42  0034 26              add     a,@r0
   43  0035 27              add     a,@r1
   44  0036 28              add     a,r0
   45  0037 29              add     a,r1
   46  0038 2A              add     a,r2
   47  0039 2B              add     a,r3
   48  003A 2C              add     a,r4
```

```
 49  003B 2D             add     a,r5
 50  003C 2E             add     a,r6
 51  003D 2F             add     a,r7
 52  003E 3027F3          jnb     badr,$-10
 53  0041 31             db      31h             ;acall
 54  0042 32             reti
 55  0043 33             rlc     a
 56  0044 3409            addc    a,#imm8
 57  0046 3535            addc    a,dadr
 58  0048 36             addc    a,@r0
 59  0049 37             addc    a,@r1
 60  004A 38             addc    a,r0
 61  004B 39             addc    a,r1
 62  004C 3A             addc    a,r2
 63  004D 3B             addc    a,r3
 64  004E 3C             addc    a,r4
 65  004F 3D             addc    a,r5
 66  0050 3E             addc    a,r6
 67  0051 3F             addc    a,r7
 68  0052 40F4            jc      $-10
 69  0054 41             db      41h             ;ajmp
 70  0055 4235            orl     dadr,a
 71  0057 433509          orl     dadr,#imm8
 72  005A 4409            orl     a,#imm8
 73  005C 4535            orl     a,dadr
 74  005E 46             orl     a,@r0
 75  005F 47             orl     a,@r1
 76  0060 48             orl     a,r0
 77  0061 49             orl     a,r1
 78  0062 4A             orl     a,r2
 79  0063 4B             orl     a,r3
 80  0064 4C             orl     a,r4
 81  0065 4D             orl     a,r5
 82  0066 4E             orl     a,r6
 83  0067 4F             orl     a,r7
 84  0068 50F4            jnc     $-10
 85  006A 51             db      51h             ;acall
 86  006B 5235            anl     dadr,a
 87  006D 533509          anl     dadr,#imm8
 88  0070 5409            anl     a,#imm8
 89  0072 5535            anl     a,dadr
 90  0074 56             anl     a,@r0
 91  0075 57             anl     a,@r1
 92  0076 58             anl     a,r0
 93  0077 59             anl     a,r1
 94  0078 5A             anl     a,r2
 95  0079 5B             anl     a,r3
 96  007A 5C             anl     a,r4
 97  007B 5D             anl     a,r5
 98  007C 5E             anl     a,r6
 99  007D 5F             anl     a,r7
100  007E 60F4            jz      $-10
101  0080 61             db      61h             ;ajmp
102  0081 6235            xrl     dadr,a
103  0083 633509          xrl     dadr,#imm8
104  0086 6409            xrl     a,#imm8
105  0088 6535            xrl     a,dadr
106  008A 66             xrl     a,@r0
```

```
107  008B 67            xrl    a,@r1
108  008C 68            xrl    a,r0
109  008D 69            xrl    a,r1
110  008E 6A            xrl    a,r2
111  008F 6B            xrl    a,r3
112  0090 6C            xrl    a,r4
113  0091 6D            xrl    a,r5
114  0092 6E            xrl    a,r6
115  0093 6F            xrl    a,r7
116  0094 70F4          jnz    $-10
117  0096 71            db     71h           ;acall
118  0097 7227          orl    c,badr
119  0099 73            jmp    @a+dptr
120  009A 7409          mov    a,#imm8
121  009C 753509        mov    dadr,#imm8
122  009F 7609          mov    @r0,#imm8
123  00A1 7709          mov    @r1,#imm8
124  00A3 7809          mov    r0,#imm8
125  00A5 7909          mov    r1,#imm8
126  00A7 7A09          mov    r2,#imm8
127  00A9 7B09          mov    r3,#imm8
128  00AB 7C09          mov    r4,#imm8
129  00AD 7D09          mov    r5,#imm8
130  00AF 7E09          mov    r6,#imm8
131  00B1 7F09          mov    r7,#imm8
132  00B3 80F4          sjmp   $-10
133  00B5 81            db     81h           ;ajmp
134  00B6 8227          anl    c,badr
135  00B8 83            movc   a,@a+pc
136  00B9 84            div    ab
137  00BA 853A35        mov    dadr,dadr+5
138  00BD 8635          mov    dadr,@r0
139  00BF 8735          mov    dadr,@r1
140  00C1 8835          mov    dadr,r0
141  00C3 8935          mov    dadr,r1
142  00C5 8A35          mov    dadr,r2
143  00C7 8B35          mov    dadr,r3
144  00C9 8C35          mov    dadr,r4
145  00CB 8D35          mov    dadr,r5
146  00CD 8E35          mov    dadr,r6
147  00CF 8F35          mov    dadr,r7
148  00D1 906162        mov    dptr,#imm16
149  00D4 91            db     91h           ;acall
150  00D5 9227          mov    badr,c
151  00D7 93            movc   a,@a+dptr
152  00D8 9409          subb   a,#imm8
153  00DA 9535          subb   a,dadr
154  00DC 96            subb   a,@r0
155  00DD 97            subb   a,@r1
156  00DE 98            subb   a,r0
157  00DF 99            subb   a,r1
158  00E0 9A            subb   a,r2
159  00E1 9B            subb   a,r3
160  00E2 9C            subb   a,r4
161  00E3 9D            subb   a,r5
162  00E4 9E            subb   a,r6
163  00E5 9F            subb   a,r7
164  00E6 A027          orl    c,/badr
```

```
165  00E8 A1             db     0a1h          ;ajmp
166  00E9 A227           mov    c,badr
167  00EB A3             inc    dptr
168  00EC A4             mul    ab
169  00ED A5             db     0a5h          ;reserved
170  00EE A635           mov    @r0,dadr
171  00F0 A735           mov    @r1,dadr
172  00F2 A835           mov    r0,dadr
173  00F4 A935           mov    r1,dadr
174  00F6 AA35           mov    r2,dadr
175  00F8 AB35           mov    r3,dadr
176  00FA AC35           mov    r4,dadr
177  00FC AD35           mov    r5,dadr
178  00FE AE35           mov    r6,dadr
179  0100 AF35           mov    r7,dadr
180  0102 B027           anl    c,/badr
181  0104 B1             db     0b1h          ;acall
182  0105 B227           cpl    badr
183  0107 B3             cpl    c
184  0108 B409F3         cjne   a,#imm8,$-10
185  010B B535F3         cjne   a,dadr,$-10
186  010E B609F3         cjne   @r0,#imm8,$-10
187  0111 B709F3         cjne   @r1,#imm8,$-10
188  0114 B809F3         cjne   r0,#imm8,$-10
189  0117 B909F3         cjne   r1,#imm8,$-10
190  011A BA09F3         cjne   r2,#imm8,$-10
191  011D BB09F3         cjne   r3,#imm8,$-10
192  0120 BC09F3         cjne   r4,#imm8,$-10
193  0123 BD09F3         cjne   r5,#imm8,$-10
194  0126 BE09F3         cjne   r6,#imm8,$-10
195  0129 BF09F3         cjne   r7,#imm8,$-10
196  012C C035           push   dadr
197  012E C1             db     0c1h          ;ajmp
198  012F C227           clr    badr
199  0131 C3             clr    c
200  0132 C4             swap   a
201  0133 C535           xch    a,dadr
202  0135 C6             xch    a,@r0
203  0136 C7             xch    a,@r1
204  0137 C8             xch    a,r0
205  0138 C9             xch    a,r1
206  0139 CA             xch    a,r2
207  013A CB             xch    a,r3
208  013B CC             xch    a,r4
209  013C CD             xch    a,r5
210  013D CE             xch    a,r6
211  013E CF             xch    a,r7
212  013F D035           pop    dadr
213  0141 D1             db     0d1h          ;acall
214  0142 D227           setb   badr
215  0144 D3             setb   c
216  0145 D4             da     a
217  0146 D535F3         djnz   dadr,$-10
218  0149 D6             xchd   a,@r0
219  014A D7             xchd   a,@r1
220  014B D8F4           djnz   r0,$-10
221  014D D9F4           djnz   r1,$-10
222  014F DAF4           djnz   r2,$-10
```

```
223  0151 DBF4            djnz    r3,$-10
224  0153 DCF4            djnz    r4,$-10
225  0155 DDF4            djnz    r5,$-10
226  0157 DEF4            djnz    r6,$-10
227  0159 DFF4            djnz    r7,$-10
228  015B E0              movx    a,@dptr
229  015C E1              db      0e1h            ;ajmp
230  015D E2              movx    a,@r0
231  015E E3              movx    a,@r1
232  015F E4              clr     a
233  0160 E535            mov     a,dadr
234  0162 E6              mov     a,@r0
235  0163 E7              mov     a,@r1
236  0164 E8              mov     a,r0
237  0165 E9              mov     a,r1
238  0166 EA              mov     a,r2
239  0167 EB              mov     a,r3
240  0168 EC              mov     a,r4
241  0169 ED              mov     a,r5
242  016A EE              mov     a,r6
243  016B EF              mov     a,r7
244  016C F0              movx    @dptr,a
245  016D F1              db      0f1h            ;acall
246  016E F2              movx    @r0,a
247  016F F3              movx    @r1,a
248  0170 F4              cpl     a
249  0171 F535            mov     dadr,a
250  0173 F6              mov     @r0,a
251  0174 F7              mov     @r1,a
252  0175 F8              mov     r0,a
253  0176 F9              mov     r1,a
254  0177 FA              mov     r2,a
255  0178 FB              mov     r3,a
256  0179 FC              mov     r4,a
257  017A FD              mov     r5,a
258  017B FE              mov     r6,a
259  017C FF              mov     r7,a
260
261  0023          padr    =       23h
262  0027          badr    =       27h
263  0035          dadr    =       35h
264  8765          ladr    =       8765h
265  6162          imm16   =       'ab'
266  0009          imm8    =       9
267
268          ;  This part shows how the CALL and JMP instructions
269          ;  are converted into ACALL, LCALL, SJMP, AJMP and LJMP.
270  017D        extern  somewhere
271
272  017D 1100             call    0
273                 ; ACALL because:
274                 ;   1.  In ASEG
275                 ;   2.  Operand contains no forward references
276                 ;   3.  Operand is absolute
277                 ;   4.  Operand is within the current 2k page
278
279  017F 120BB8            call    3000
280                 ; LCALL because:
```

```
281                ;   1.  Operand is not within the current 2k page
282
283  0182 120000           call    somewhere
284                ; LCALL because:
285                ;   1.  Operand is external
286
287  0185 120188           call    forward
288  0188           forward:
289                ; LCALL because:
290                ;   1.  Operand contains a forward reference
291
292  0188 0100             jmp     0
293                ; AJMP because:
294                ; 1.  In ASEG
295                ; 2.  Operand contains no forward references
296                ; 3.  Operand is absolute
297                ; 4.  Operand is within the current 2k page
298                ; 5.  Operand is more than -126 to +129 bytes from $
299
300  018A 8008             jmp     $+10
301                ; SJMP because:
302                ;   2.  Operand contains no forward references
303                ;   3.  Operand has the type of the current segment
304                ;   3.  Operand is within -126 to +129 bytes from $
305  0000                  rseg    anywhere
306
307  0000 12000A           call    $+10
308                ; LCALL because:
309                ;   1.  In a relocatable section
310
311                ; ICC8051 4.00 operators
312                ;
313  0003                  extern c_func1
314  0003                  $deffn c_func1(0,5,0,0),f,
315                g
316  0003                  $deffn g(1,1,3,4,5,6,7,8)
317  0003                  $deffn f(8,4,0,0,0,0,0,6),g
318  0000                  rseg    code
319  0000           f:
320  0000 E502              mov     a,$locbd f
321  0002 E502              mov     a,$locbi f + 4
322  0004 E502              mov     a,$locbb f - 1
323  0006 7400              mov     a,#$byte3 f
324  0008 7400              mov     a,#$byte3 c_func1
325  000A 7412              mov     a,#$byte3 123456H
326  000C 900206            mov     dptr,#$locbx f + 3
327  000F E502              mov     a,$prmbd f
328  0011 E502              mov     a,$prmbi f + 4
329  0013 E502              mov     a,$prmbb f - 1
330  0015 900202            mov     dptr,#$prmbx f + 3
331  0018 E502              mov     a,$locbd f
332  001A E500              mov     a,$locbi c_func1 + 4
333  001C E500              mov     a,$locbb c_func1 - 1
334  001E 900006            mov     dptr,#$locbx c_func1 + 3
335  0021 E502              mov     a,$prmbd f
336  0023 E500              mov     a,$prmbi c_func1 + 4
337  0025 E500              mov     a,$prmbb c_func1 - 1
338  0027 900002            mov     dptr,#$prmbx c_func1 + 3
```

```
339  002A 850100          mov    $prmbd c_func1 - 3,$prmbd g + 3
340  002D 750002          mov    $prmbd c_func1 - 3,#$prmbd f + 3
341  0030 E501            mov    a,high ($locbx g + 5)
342  0032 E501            mov    a,low ($locbx g + 5)
343  0034 E501            mov    a,high ($prmbx g + 5)
344  0036 E501            mov    a,low ($prmbx g + 5)
345  0038 7400            mov    a,#low $ifref c_func1
346  003A 7401            mov    a,#high $ifref g
347  003C 0206            dw     ($locbx f + 5)
348  003E 05              db     low(f + 5)
349  003F 127B02          lcall   $reffn f
350  0042           g:
351  0042 025B            dw     $ifref f + 6
352  0044 005B            dw     $ifref c_func1
353  0046                 end
```