Apêndice B
Conjunto de Instruções do 8051

A.G. e E.T.M. / 2001 (revisão)

O 8051 apresenta 111 tipos de instruções, sendo 49 de um byte, 45 de dois bytes e 17 de três bytes. Levando-se em contas as variações de cada tipo, chega-se a 225 instruções, onde os *opcodes* estão entre 00h e FFh, exceto A5.

A seguir, é apresentado um resumo das instruções.

1. Instruções Aritméticas

São as instruções usuais de um operador de 8 bits: soma (ADD), soma com carry (ADC), subtração com borrow (SBB), incremento (INC), decremento (DEC) e ajuste decimal (DA A). Apresenta também duas operações que não são típicas de microprocessadores de 8 bits: multiplicação (MUL AB) e divisão (DIV AB).

- ADD

Adiciona ao acumulador uma variável de um byte. O resultado da operação é armazenado no acumulador. O flag C é ativado se há um overflow no bit 7, caso contrário permanece em 0. O flag AC é ativado se houver um "vai-um" do bit 3.

- ADDC

A instrução ADDC realiza a soma entre o conteúdo do acumulador, do valor da variável de um byte e o conteúdo do flag de carry.

- SUBB

Esta instrução subtrai do acumulador, o valor da variável de um byte e o conteúdo do Borrow do acumulador. O resultado é armazenado no acumulador. O flag de carry indica que houve um "empresta um" durante a subtração, ou seja um número maior foi subtraído de um menor. Nas demais situações, o flag de carry permanece em zero.

Flag OV**:** Este flag é ativado se ocorre "vai-um" no bit 6 e não ocorre "vai-um" no bit 7, ou se ocorre "vai-um" no bit 7 e não ocorre "vai-um" no bit 6. Na adição com os operadores representados com sinal, a ativação deste flag indica portanto, que o resultado da adição de dois números positivos é negativo. Também indica que um resultado negativo é obtido quando um número negativo é subtraído de um número positivo, ou um resultado positivo é obtido quando um número positivo é subtraído de um número negativo (overflow).

- DA A

Instrução de ajuste decimal do acumulador. O valor de oito bits no acumulador é convertido em dois dígitos BCD de quatro bits cada. Se o conteúdo dos bits 0-3 for maior que 9, ou o flag AC estiver ativado, então o valor 6 é adicionado ao conteúdo do acumulador para a obtenção do correto código BCD. O flag de carry não é afetado. Exemplo de utilização da instrução:

```
MOV R0, #23     ; R0 ←23
MOV A, #47      ; A ← 47
ADD A, R0       ; A ← 6A (onde 6A=23h+47h)
DA A            ; A ←70
```

- MUL AB

Esta instrução multiplica o conteúdo do acumulador pelo conteúdo do registrador B, ambos como inteiros, sem sinal. O byte menos significativo do resultado é armazenado no acumulador e o byte mais significativo é armazenado no registrador B. Se o byte de maior ordem do resultado for zero, o flag OV também é zero, caso contrário é ativado, indicando que o resultado apresenta mais que oito bits. Portanto, o flag OV pode ser utilizado pelo programador para determinar se o registrador B dever ser processado após a instrução.

- DIV AB

Esta instrução faz com que o conteúdo do acumulador seja dividido pelo conteúdo de B, ambos representados como números inteiros sem sinal. A parte inteira do resultado é armazenada no acumulador. O resto é armazenado no registrador B. Se o conteúdo do registrador B anterior à operação de divisão for zero, o flag OV é ativado, caso contrário é zero.

- INC

Soma 1 ao conteúdo do registrador ou da posição de memória endereçada na instrução. Flags não são alterados por esta instrução. Exemplo:

```
INC DPTR; incrementa o conteúdo do registrador de ponteiro de dados.
INC 23h;  incrementa o conteúdo da memória RAM de endereço 23h.
```

- DEC

Subtrai 1 do conteúdo do registrador ou da posição de memória endereçados na instrução. Não afeta flags. Exemplo:

```
DEC 23h; subtrai 1 do conteúdo da memória RAM de endereço 23h
```

2. Instruções Lógicas

Este grupo de instruções realiza as funções lógicas E (ANL), OU (ORL) e OU-EXCLUSIVO (XRL) entre as duas variáveis endereçadas pelas instruções. O resultado é armazenado na primeira variável, não se restringindo portanto, ao acumulador. As instruções para tornar zero (CLR), complementar (CPL), bem como as instruções de deslocamento (RL, RLC, RR e RRC) também fazem parte deste grupo.

A instrução SWAP A troca os quatro bits menos significativos do acumulador, pelos quatro bits mais significativos. Equivale a quatro RR A ou a quatro RL A.

Nenhuma das instruções contidas neste grupo afeta os flags, exceto RRC e RLC que realizam o deslocamento, utilizando-se do flag de carry.

As instruções lógicas são muito empregadas para realizar o clear, o set e o complemento de um ou mais bits da RAM, das portas de saída ou dos registradores de controle, através da utilização das instruções lógicas ANL, ORL e XRL. Exemplos:

```
ANL P1, #11100000B      ;realiza um clear nos bits 4-0 da porta 1.
ORL P1, A               ;seta os pinos de P1 segundo os valores dos bits do acumulador
XRL A, #0FFH            ;complementa os bits do acumulador.
```

3. Instruções de Transferência de Dados

A instrução básica de transferência de dados é mover, realizada pelas instruções MOV, MOVC e MOVX. Também estão incluídas neste grupo as instruções PUSH e POP, referentes ao ponteiro de pilha, bem como a instrução XCH.

- MOV

Esta instrução é utilizada para referenciar a memória RAM interna e o espaço dos registradores de função especial (SFR). Exemplos:

```
MOV A, 20h      ;move para o acumulador o dado presente no endereço 20h.
MOV @R1, #32h ;move para a posição endereçada indiretamente pelo registrador R1, o valor
                ;32h.
```

- MOVX

É utilizada para referenciar a memória RAM externa. Exemplo:

```
MOVX B, @R2   ;move para o registrador B o dado presente na posição da RAM externa
                ;endereçada por R1.
```

- MOVC A, @A+DPTR **e** MOVC A, @A+PC

Estas instruções armazenam no acumulador bytes dos dados do espaço da memória de programa. Ambas utilizam a forma de endereçamento indexado, cujo endereço efetivo é a soma dos conteúdos do acumulador e do registrador de ponteiro de dados DPTR ou contador de programas PC, respectivamente.

- XCH

Troca o conteúdo do acumulador com o conteúdo da memória ou registrador endereçados na instrução.

- XCHD

Troca os quatro bits menos significativos do conteúdo do acumulador com os quatro bits menos significativos do conteúdo da posição de memória ou registrador endereçados na instrução.

- PUSH

O valor do apontador de pilha (stack pointer) é incrementado de uma unidade e depois o conteúdo da posição de memória endereçada por modo direto na instrução é escrito na pilha.

- POP

O conteúdo da posição endereçada pelo apontador de pilha (Stack Pointer) é lido e transferido para a posição de memória endereçada por modo direto na instrução e o conteúdo do stack pointer é decrementado de uma unidade.

4. Instruções de Manipulação de Variáveis Booleanas

Este grupo inclui as instruções que permitem setar e zerar bits (CLR e SETB), complementar (CPL) e realizar operações lógicas E (ANL C, bit) e OU (ORL C, bit) entre qualquer bit e o carry. Também se encontram instruções de desvios condicionais que testam valores dos bits endereçáveis ou do flag de carry.

Nos mnemônicos das instruções booleanas relacionadas ao carry utiliza-se o símbolo C.

- JC/JNC

Estas instruções testam se o flag de carry do registrador PSW está setado ou resetado. Se a condição for verdadeira, o desvio é realizado.

- JB/JNB/JBC

Estas instruções testam se o bit endereçado está setado (JB) ou resetado (JNB), sendo que a instrução JBC zera o bit depois do desvio. Exemplo:

```
LP1:    JB P3.4, T0_ON  ;testa se  a entrada do temporizador 0 está setado
        SJMP LP1        ;sendo falso, retorna para LP1, sendo verdadeiro, prosseguem em
T0_ON: ……              ;T0_ON;
```

- CLR C

Torna zero o flag de carry.

5. Desvios Incondicionais

- LJMP

Esta é a instrução de desvio longo que usa endereços de 16 bits como parte da instrução, gerando instruções de 3 bytes. Possibilita acessar qualquer localização no espaço de 64Kb de memória de programa. O formato da instrução é:

opcode	Addr15-addr8	addr7- addr0

Exemplo:

```
TEMP    EQU   1000H
ORG     1000H
TEMP:   LJMP LABEL1
LABEL1: LJMP  TEMP
```

- AJMP

Esta instrução desvia o fluxo do programa para outro endereço numa faixa de 2Kbytes referentes à atual posição da memória de programa. O formato da instrução é:

ad10-ad8 / opcode	Addr7-addr0

Exemplo:

```
        AJMP NEXT;
        MOV A, #01H;
NEXT:   MOV A, #0FEH;
```

- SJMP

Desvio curto relativo. Usa um offset de oito bits, podendo ser negativo ou positivo, ou seja o endereço de desvio pode ser de até 128 bytes a menos ou 127 bytes a mais, respectivamente, em relação à posição atual da memória. O formato da instrução é:

opcode	Offset relativo

Exemplo:

```
        SJMP 00h;       ;desvia para a próxima instrução (offset = 0)
NEXT:   SJMP 0FEH;      ;retorna para a instrução anterior (offset = -1)
```

- JMP @A+DPTR

O conteúdo do PC é armazenado com o valor da soma dos conteúdos dos registradores DPTR e acumulador, prosseguindo portanto a execução do programa a partir deste ponto.

6. Desvios Condicionais

- JZ/JNZ

Estas instruções desviam para o endereço especificado, se o conteúdo do acumulador for igual a ou diferente de zero, respectivamente

Exemplo:

```
        MOV A, #00h
        JNZ LB2
LB1:    JZ LB3
LB2:    MOV A, #0FFh
LB3:    SJMP LB3
```

- JC /JNC

Desvia se o flag de carry está ativado ou desativado respectivamente.

7. Sub-rotina

- ACALL

Call absoluto. É utilizada para acessar sub-rotinas em endereços dentro de um espaço de até 2K com relação ao endereço armazenado em PC.

- LCALL

Chamada de sub-rotina com formato longo. Permite acessar os 64K da memória de programa.

- RET

Retorno de sub-rotina. O endereço de retorno é recuperado a partir da pilha e então armazenado no PC. O valor do registrador SP é decrementado de 2.

- RETI

Quando ocorre uma interrupção, o processador executa uma instrução LCALL para a rotina de atendimento àquela interrupção. Utilizando-se RETI, em vez de RET é possível habilitar interrupções de mesma prioridade, ou menos prioritárias já pendentes a serem executadas. Esta instrução não recupera o conteúdo do acumulador e do registrador PSW, assim como as interrupções não salvam os mesmos. Estas tarefas devem ser previstas no software.

8. Instruções que combinam Operações e Desvios

As seguintes instruções realizam operações com operandos de um byte e realizam o desvio condicionado ao resultado.

- CJNE

Compara dois operandos e desvia se forem diferentes. O flag de carry é afetado da mesma forma que na operação de subtração.

- DJNZ

Decrementa o dado presente no registrador ou o dado na posição de memória diretamente endereçada e desvia se o resultado da operação for diferente de zero, sem afetar quaisquer flags.

Resumo das Instruções do 8051

(baseado no manual da Intel)

Antes de passarmos a apresentar o conjunto completo das instruções do 8051, convém apresentarmos uma visão geral de sua estrutura interna. A figura abaixo ilustra o diagrama de blocos da arquitetura interna do microcontrolador 8051.

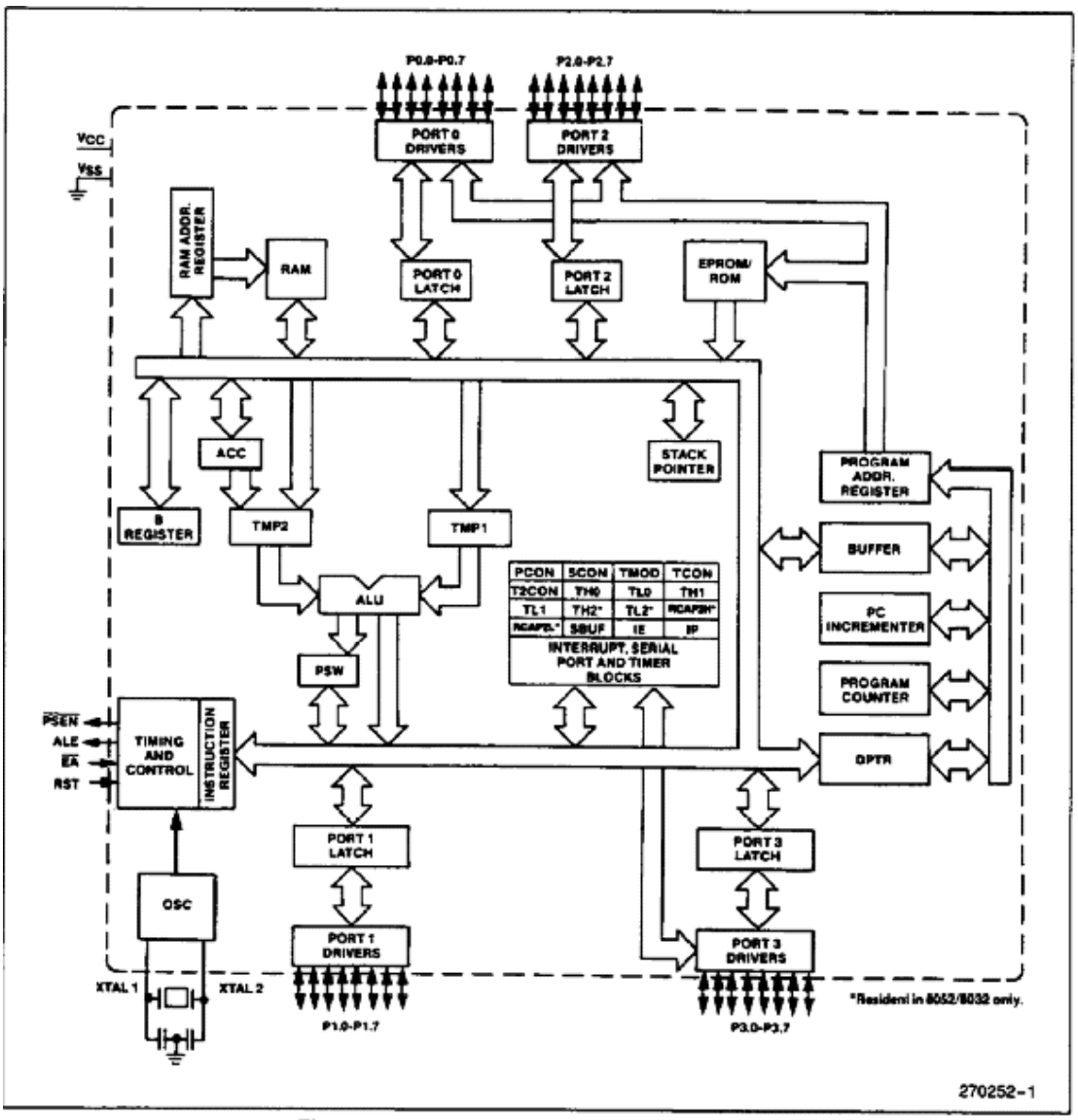

Figure 1. MCS-51 Architectural Block Diagram

A figura mostra os principais componentes internos (como por exemplo, as portas de E/S, o módulo de temporização e controle, a memória interna) e os seus registradores (como por exemplo, ACC, PC, DPTR, PSW).

Segue abaixo um resumo das instruções do 8051 (em inglês).

Instructions that Affect Flag Settings:

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

Note that operations on SFR byte address 208 or bit addresses 209-215 (i.e., the PSW or bits in the PSW) will also affect flag settings.

Note on instruction set and addressing modes:

Rn - Register R7 – R0 of currently selected Register Bank.

direct - 8-bit internal data location's address. This could be an Internal Data RAM location (0-127) or a SFR [i.e., I/O port, control register, status register, etc. (128-255)].

@Ri - 8-bit internal data RAM location (0-255) addressed indirectly through register R1 or R0.

#data - 8-bit constant included in instruction.

#data 16 - 16-bit constant included in instruction.

addr 16 - 16-bit destination address. Used by LCALL & LJMP. A branch can be anywhere within 64K-byte Program Memory address space.

addr 11 - 11-bit destination address. Used by ACALL & AJMP. The branch will be within the same 2K-byte page of program memory as the first byte of the following instruction.

rel - Signed (two's complement) 8-bit offset byte. Used by SJMP and all conditional jumps. Range is -128 to +127 bytes relative to the first byte of the following instruction.

bit - Direct Addressed bit in Internal Data RAM or Special Function Register.

Arithmetic Operations:

Mnemonic	Description	Byte	Oscillator Period
ADD A, Rn	Add register to accumulator	1	12
ADD A, direct	Add direct byte to accumulator	2	12
ADD A, @Ri	Add indirect RAM to accumulator	1	12
ADD A, #data	Add immediate data to accumulator	2	12
ADDC A, Rn	Add register to accumulator with Carry	1	12
ADDC A, direct	Add direct byte to accumulator with Carry	2	12
ADDC A, @Ri	Add indirect RAM to accumulator with Carry	1	12
ADDC A, #data	Add immediate data to accumulator with Carry	2	12
SUBB A, Rn	Subtract register from accumulator with borrow	1	12
SUBB A, direct	Subtract direct byte from accumulator with borrow	2	12
SUBB A, @Ri	Subtract indirect RAM from accumulator with borrow	1	12
SUBB A, #data	Subtract immediate data from accumulator with borrow	2	12
INC A	Increment accumulator	1	12
INC Rn	Increment register	1	12
INC direct	Increment direct byte	2	12
INC @Ri	Increment indirect RAM	1	12
DEC A	Decrement accumulator	1	12
DEC Rn	Decrement register	1	12
DEC direct	Decrement direct byte	2	12
DEC @Ri	Decrement indirect RAM	1	12
INC DPTR	Increment Data Pointer	1	24
MUL A,B	Multiply A & B	1	48
DIV A,B	Divide A by B	1	48
DA A	Decimal Adjust Accumulator	1	12

Logical Operations

Mnemonic	Description	Byte	Oscillator Period
ANL A, Rn	AND Register to Accumulator	1	12
ANL A, direct	AND direct byte to Accumulator	2	12
ANL A, @Ri	AND indirect RAM to Accumulator	1	12
ANL A, #data	AND immediate data to Accumulator	2	12
ANL direct, A	AND Accumulator to direct byte	2	12
ANL direct, #data	AND immediate data to direct byte	3	24
ORL A, Rn	OR Register to Accumulator	1	12
ORL A, direct	OR direct byte to Accumulator	2	12
ORL A, @Ri	OR indirect RAM to Accumulator	1	12
ORL A, #data	OR immediate data to Accumulator	2	12
ORL direct, A	OR Accumulator to direct byte	2	12
ORL direct, #data	OR immediate data to direct byte	3	24
XRL A, Rn	Exclusive-OR Register to Accumulator	1	12
XRL A, direct	Exclusive-OR direct byte to Accumulator	2	12
XRL A, @Ri	Exclusive-OR indirect RAM to Accumulator	1	12
XRL A, #data	Exclusive-OR immediate data to Accumulator	2	12
XRL direct, A	Exclusive-OR Accumulator to direct byte	2	12
XRL direct, #data	Exclusive-OR immediate data to direct byte	3	24
CLR A	Clear Accumulator	1	12
CPL A	Complement Accumulator	1	12
RL A	Rotate Accumulator Left	1	12
RLC A	Rotate Accumulator Left through the Carry	1	12
RR A	Rotate Accumulator Right	1	12
RRC A	Rotate Accumulator Right through the Carry	1	12
SWAP A	Swap nibbles within Accumulator	1	12

Data Transfer Operations

Mnemonic	Description	Byte	Oscillator Period
MOV A, Rn	Move register to Accumulator	1	12
MOV A, direct	Move direct byte to Accumulator	2	12
MOV A, @Ri	Move indirect RAM to Accumulator	1	12
MOV A, #data	Move immediate data to Accumulator	2	12
MOV Rn, A	Move Accumulator to register	1	12
MOV Rn, direct	Move direct byte to register	2	24
MOV Rn, #data	Move immediate data to register	2	12
MOV direct, A	Move Accumulator to direct byte	2	12
MOV direct, Rn	Move register to direct byte	2	24
MOV direct, direct	Move direct byte to direct byte	3	24
MOV direct, @Ri	Move indirect RAM to direct byte	2	24
MOV direct, #data	Move immediate data to direct byte	3	24
MOV @Ri, A	Move Accumulator to indirect RAM	1	12
MOV @Ri, direct	Move direct byte to indirect RAM	2	24
MOV @Ri, #data	Move immediate data to indirect RAM	2	12
MOV DPTR, #data16	Load Data Pointer with a 16-bit constant	3	24
MOVC A, @A+DPTR	Move code byte relative to DPTR to Accumulator	1	24
MOVC A, @A+PC	Move code byte relative to PC to Accumulator	1	24
MOVX A, @Ri	Move External RAM (8-bit addr) to Accumulator	1	24
MOVX A, @DPTR	Move External RAM (16-bit addr) to Accumulator	1	24
MOVX @Ri, A	Move Accumulator to External RAM (8-bit addr)	1	24
MOVX @DPTR, A	Move Accumulator to External RAM (16-bit addr)	1	24
PUSH direct	Push direct byte onto stack	2	24
POP direct	Pop direct byte from stack	2	24
XCH A, Rn	Exchange register with Accumulator	1	12
XCH A, direct	Exchange direct byte with Accumulator	2	12
XCH A, @Ri	Exchange indirect RAM with Accumulator	1	12
XCHD A, @Ri	Exchange low-order Digit indirect RAM with Acc	1	12

Boolean Variable Manipulation

Mnemonic	Description	Byte	Oscillator Period
CLR C	Clear Carry	1	12
CLR bit	Clear direct bit	2	12
SETB C	Set Carry	1	12
SETB bit	Set direct bit	2	12
CPL C	Complement Carry	1	12
CPL bit	Complement direct bit	2	12
ANL C, bit	AND direct bit to Carry	2	24
ANL C,/bit	AND complement direct bit to Carry	2	24
ORL C, bit	OR direct bit to Carry	2	24
ORL C,/bit	OR complement direct bit to Carry	2	24
MOV C, bit	Move direct bit to Carry	2	12
MOV bit, C	Move Carry to direct bit	2	24
JC rel	Jump if Carry is set	2	24
JNC rel	Jump if Carry is not set	2	24
JB bit, rel	Jump if direct bit is set	3	24
JNB bit, rel	Jump if direct bit is not set	3	24
JBC bit, rel	Jump if direct bit is set & clear bit	3	24

Program Branching Operations

Mnemonic	Description	Byte	Oscillator Period
ACALL addr11	Absolute Subroutine Call	2	24
LCALL addr16	Long Subroutine Call	3	24
RET	Return from subroutine	1	24
RETI	Return from interrupt	1	24
AJMP addr11	Absolute Jump	2	24
LJMP addr16	Long Jump	3	24
SJMP rel	Short Jump (relative addr)	2	24
JMP @A+DPTR	Jump indirect relative to the DPTR	1	24
JZ rel	Jump if Accumulator is zero	2	24
JNZ rel	Jump if Accumulator is not zero	2	24
CJNE A,direct,rel	Compare direct byte to Acc and jump if not equal	3	24
CJNE A,#data,rel	Compare immediate to Acc and jump if not equal	3	24
CJNE Rn,#data,rel	Compare immediate to register and jump if not equal	3	24
CJNE @Ri,#data,rel	Compare immediate to indirect and jump if not equal	3	24
DJNZ Rn, rel	Decrement register and jump if not zero	2	24
DJNZ direct, rel	Decrement direct byte and jump if not zero	3	24
NOP	No Operation	1	12