Exemplos de programas em assembly do 8051

António Abreu
Departamento de Engenharia Electrotécnica
Escola Superior de Tecnologia de Setúbal

May 3, 2007

Turns on a LED and Loops forever

```
org 0;  Execution Starts Here
Loop:;  Loop Here Forever
        mov C,P1.1; Get the button State
        mov P1.0,C; Reflect it in the LED
        sjmp Loop
```

Debouncing a button

```
; This Application polls a button and toggles a LE
; when the button is pressed (and has been "Debour
; The button is assumed to be debounced if it does
; state for 20 msecs.
;  P1.0 is the LED (to light, it is pulled down)
;  P1.1 is connected to a Momentary ON Switch that
org 0;   Execution starts here
Loop: ;  Loop here forever
PressWait:; Wait for the button to be pressed
        jb P1.1,PressWait
        mov A,#191;  Setup 20 msec delay
        mov B,#10
PressDebounce:; Now, see if the button is held dow
        jb P1.1,PressWait; 20 msec - if released,
        djnz ACC,PressDebounce
        djnz B,PressDebounce
;  #### - Button press is debounced
        cpl P1.0;  Toggle the LED state
ReleaseWait:;  Now, wait for the button to be rele
        jnb P1.1,ReleaseWait
        sjmp Loop
```

Serial I/O

```
; initialize registers for setting up serial commu
poweron:
        mov     SP,#0x30 ; move stack memory to a
        clr     PSW.3 ;set for register bank 0 (in
        clr     PSW.4
        orl     PCON,#10000000b ; set double baud
        mov     TMOD,#00010001b
        mov     SCON,#01010000b ; Set Serial for m
        orl     TCON,#01010010b ; Start timer 1
        mov     A,#baud_const
        mov     TH1,A
        setb    TI ;TI is normally set in this pro
        clr     RI ;RI is normally cleared
        ;jump to main program from here...
```

```
; for receiving a byte through the COM port
cin:    jnb     RI, cin
        clr     RI
        mov     A, SBUF
        ret
```

```
; for sending a byte through the COM port
cout:    jnb     TI, cout
         clr     TI
         mov     SBUF, A
         ret
newline:push   ACC
         mov     A, #13
         acall   cout
         mov     A, #10
         acall   cout
         pop     ACC
         ret
```

Implementação de um if-then-else em C

```
If A==0 {
  L1;
  L2;
}
else {
  L3;
  L4;
}
L5;
```

Em ASM51

```
        cjne A,#0,else
        L1
        L2
        jmp cont
else:
        L3
        L4
cont:
        L5
```

Implementação de um switch em C

```
switch(A) {
  case 0:
    L1;
    L2;
    break;
  case 1:
    L3;
    L4;
    break;
  case 2:
    L5;
    L6;
    break;
};
```

Em ASM51

O programa principal determina o valor de A e chama a rotina SWITCH. Cuidado com o valor de A, nomeadamente não pode ser maior que 128 e não deve tomar um valor não esperado; no caso presente só deve tomar os valores 0, 1 e 2.

```
switch:
        mov B,#2 ; 2 e' o espaco ocupado pelo salt
        mul AB
        mov DPTR,#cases
        jmp @A+DPTR
cases:
        ajmp case0
        ajmp case1
        ajmp case2
case0:
        L1
        L2
        ret
case1:
        L3
        L4
        ret
case2:
        L5
        L6
        ret
```

Relação $A > B$

Faz-se A-B e consulta-se o bit de carry. Seja c(A-B) o carry da subtracção de A por B. Se c(A-B)=1, então $A > B$ é falso. Se c(A-B)=0, pode acontecer que $A = B$, e nesse caso $A > B$ é falso, ou pode acontecer que $A \neq B$, e nesse caso $A > B$ é verdadeiro.

ASM51

O resultado é retornado na flag C: C=1 significa $A > B$, e C=0 significa $A \leq B$.

```
ABIGB:
            push 20H
            mov 20H,B
            clr C ; devido a subb
            subb A,20H
            pop 20H
            jc lessequal
            jz lessequal
            setb C
            ret
lessequal:
            clr C
            ret
```

MOVC

Imprimir em LCD uma string escrita no endereço MSG. Omitem-se os pormenores do LCD.

```
            MOV DPTR,#MSG
            MOV R0,#0
MORE:       MOV A,R0
            MOVC A,@A+DPTR
            JZ ENDSTRING
            INC R0
            CALL LCDCHAR
            JMP MORE
ENDSTRING: ... MSG:        'Hello world'0 ; 0 \'e c
string
```

Endereçamento indirecto

Copiar o bloco de memória com início no endereço XH, com tamanho Y, para a posição a partir de ZH.

```
          mov R0,#XH
          mov R1,#ZH
          mov R2,#Y
          call copy_mem
          ...
copy_mem:
          push A
onemore:  mov A,@R0
          mov @R1,A
          inc R0
          inc R1
          djnz R2,onemore
          pop A
          ret
```

Cópia de bloco de memória, em C

```
{
  char X[20];
  char Z[20];
  unsigned char Y;
  Z[0]=?;
  ...
  Z[19]=?;
  copy_mem_block(X,Z,Y);
  ...
}

copy_mem_block(char* y,char* x,unsigned char n)
{
  unsigned char i;
  for(i=0;i<n;i++)
    y[i]=x[i];
}
```