MC404

ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

2010

Prof. Paulo Cesar Centoducatte
Prof. Mario Lúcio Côrtes
Prof. Ricardo Pannain

MC404

ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

"Diretivas de Montagem"

Diretivas de Montagem
Sumário

- **Diretivas do Montador**
  - Segmento
  - Reserva de bytes e words
  - Definição de constantes
  - Definição de síbolos para registradores

Arquivo Fonte

- **[Label:] diretiva [operandos] [comentários]**
- **[Label:] instrução [operandos] [comentários]**
- **Comentários**
- **Linha em branco**

- **Comentários:**

  ; [texto]

Diretivas do Montador

Directive	Description
BYTE	Reserve byte to a variable
CSEG	Code Segment
DB	Define constant byte(s)
DEF	Define a symbolic name on a register
DEVICE	Define which device to assemble for
DSEG	Data Segment
DW	Define constant word(s)
ENDMACRO	End macro
EQU	Set a symbol equal to an expression
ESEG	EEPROM Segment
EXIT	Exit from file
INCLUDE	Read source from another file
LIST	Turn listfile generation on
LISTMAC	Turn macro expansion on
MACRO	Begin macro
NOLIST	Turn listfile generation off
ORG	Set program origin
SET	Set a symbol to an expression

SRAM/DSEG

Progr ou EEPROM/CSEG ou ESEG

Progr ou EEPROM/CSEG ou EEPROM

Label ← constante (não alterável)

Label ← constante (alterável)

Diretivas do Montador

```
label:    .EQU var1=100      ; Set var1 to 100 (Directive)
          .EQU var2=200      ; Set var2 to 200
test:     rjmp  test         ; Infinite loop (Instruction)
                             ; Pure comment line
                             ; Another comment line
```

Diretivas do Montador
Segmentos

- **.CSEG ; Segmento de Memória na Flash**
- **.DSEG ; Segmento de Memória na SRAM**
- **.ESEG ; Segmento de Memória na EEPROM**

Reserva de Byte(s) na SRAM

```
.DSEG
        var1:   .BYTE 1          ; reserve 1 byte to var1
        table:  .BYTE tab_size   ; reserve tab_size bytes
.CSEG
        ldi     r30,low(var1)    ; Load Z register low
        ldi     r31,high(var1)   ; Load Z register high
        ld      r1,Z             ; Load VAR1 into register 1
```

Reserva Byte(s) com inicialização

```
.CSEG
        consts: .DB 0, 255, 0b01010101, -128, 0xaa
.ESEG
        eeconst:.DB 0xff
```

Nome Simbólicos para Registradores

Syntax:

```
.DEF Symbol=Register
```

Example:

```
.DEF temp=R16
.DEF ior=R0
.CSEG
        ldi     temp,0xf0       ; Load 0xf0 into temp register
        in      ior,0x3f        ; Read SREG into ior register
        eor     temp,ior        ; Exclusive or temp and ior
```

Reserva Word(s) com Inicialização

Syntax:

```
LABEL: .DW expressionlist
```

Example:

```
.CSEG
        varlist:.DW 0,0xffff,0b1001110001010101,-32768,65535
.ESEG
        eevar:  .DW 0xffff
```

Syntax:

```
.EQU label = expression
```

Example:

```
.EQU io_offset = 0x23
.EQU porta = io_offset + 2
.CSEG                          ; Start code segment
        clr     r2             ; Clear register 2
        out     porta,r2       ; Write to Port A
```

Syntax:

```
.ORG expression
```

Example:

```
.DSEG                          ; Start data segment
.ORG 0x67                      ; Set SRAM address to hex 67
       variable:.BYTE 1        ; Reserve a byte at SRAM
                               ; adr.67H
.ESEG                          ; Start EEPROM Segment
.ORG 0x20                      ; Set EEPROM location
                               ; counter
       eevar:   .DW 0xfeff     ; Initialize one word
.CSEG
.ORG 0x10                      ; Set Program Counter to hex
                               ; 10
       mov      r0,r1          ; Do something
```

Expressões

Expressions

The Assembler incorporates expressions. Expressions can consist of operands, operators and functions. All expressions are internally 32 bits.

Operands

The following operands can be used:

- User defined labels which are given the value of the location counter at the place they appear.
- User defined variables defined by the SET directive
- User defined constants defined by the EQU directive
- Integer constants: constants can be given in several formats, including
  a) Decimal (default): 10, 255
  b) Hexadecimal (two notations): 0x0a, $0a, 0xff, $ff
  c) Binary: 0b00001010, 0b11111111
- PC - the current value of the Program memory location counter

Operadores

Logical Not	Symbol:	!
	Description:	Unary operator which returns 1 if the expression was zero, and returns 0 if the expression was nonzero
	Precedence:	14
	Example:	`ldi r16,!0xf0 ; Load r16 with 0x00`
Bitwise Not	Symbol:	~
	Description:	Unary operator which returns the input expression with all bits inverted
	Precedence:	14
	Example:	`ldi r16,~0xf0 ; Load r16 with 0x0f`
Unary Minus	Symbol:	-
	Description:	Unary operator which returns the arithmetic negation of an expression
	Precedence:	14
	Example:	`ldi r16,-2 ; Load -2(0xfe) in r16`
Multiplication	Symbol:	*
	Description:	Binary operator which returns the product of two expressions
	Precedence:	13
	Example:	`ldi r30,label*2 ; Load r30 with label*2`

Operadores (2)

Division

Symbol: /

Description: Binary operator which returns the integer quotient of the left expression divided by the right expression

Precedence: 13

Example:
```
ldi    r30,label/2     ; Load r30 with label/2
```

Addition

Symbol: +

Description: Binary operator which returns the sum of two expressions

Precedence: 12

Example:
```
ldi    r30,c1+c2       ; Load r30 with c1+c2
```

Subtraction

Symbol: -

Description: Binary operator which returns the left expression minus the right expression

Precedence: 12

Example:
```
ldi    r17,c1-c2       ;Load r17 with c1-c2
```

Shift left

Symbol: <<

Description: Binary operator which returns the left expression shifted left a number of times given by the right expression

Precedence: 11

Example:
```
ldi    r17,1<<bitmask ;Load r17 with 1 shifted
                      ;left bitmask times
```

Operadores (3)

Shift right

Symbol: >>

Description: Binary operator which returns the left expression shifted right a number of times given by the right expression.

Precedence: 11

Example:
```
ldi    r17,c1>>c2      ;Load r17 with c1 shifted
                       ;right c2 times
```

Less than

Symbol: <

Description: Binary operator which returns 1 if the signed expression to the left is Less than the signed expression to the right, 0 otherwise

Precedence: 10

Example:
```
ori    r18,bitmask*(c1<c2)+1       ;Or r18 with
                                   ;an expression
```

Less or Equal

Symbol: <=

Description: Binary operator which returns 1 if the signed expression to the left is Less than or Equal to the signed expression to the right, 0 otherwise

Precedence: 10

Example:
```
ori    r18,bitmask*(c1<=c2)+1      ;Or r18 with
                                   ;an expression
```

Greater than

Symbol: >

Description: Binary operator which returns 1 if the signed expression to the left is Greater than the signed expression to the right, 0 otherwise

Precedence: 10

Example:
```
ori    r18,bitmask*(c1>c2)+1       ;Or r18 with
                                   ;an expression
```

Operadores (4)

Greater or Equal

Symbol: >=

Description: Binary operator which returns 1 if the signed expression to the left is Greater than or Equal to the signed expression to the right, 0 otherwise

Precedence: 10

Example:
```
ori    r18,bitmask*(c1>=c2)+1      ;Or r18 with
                                   ;an expression
```

Equal

Symbol: ==

Description: Binary operator which returns 1 if the signed expression to the left is Equal to the signed expression to the right, 0 otherwise

Precedence: 9

Example:
```
andi   r19,bitmask*(c1==c2)+1      ;And r19 with
                                   ;an expression
```

Not Equal

Symbol: !=

Description: Binary operator which returns 1 if the signed expression to the left is Not Equal to the signed expression to the right, 0 otherwise

Precedence: 9

Example:
```
.SET flag=(c1!=c2)          ;Set flag to 1 or 0
```

Bitwise And

Symbol: &

Description: Binary operator which returns the bitwise And between two expressions

Precedence: 8

Example:
```
ldi    r18,High(c1&c2)     ;Load r18 with an expression
```

Operadores (5)

Bitwise Xor	Symbol:	^
	Description:	Binary operator which returns the bitwise Exclusive Or between two expressions
	Precedence:	7
	Example:	`ldi r18,Low(c1^c2) ;Load r18 with an expression`
Bitwise Or	Symbol:	|
	Description:	Binary operator which returns the bitwise Or between two expressions
	Precedence:	6
	Example:	`ldi r18,Low(c1|c2) ;Load r18 with an expression`
Logical And	Symbol:	&&
	Description:	Binary operator which returns 1 if the expressions are both nonzero, 0 otherwise
	Precedence:	5
	Example:	`ldi r18,Low(c1&&c2) ;Load r18 with an expression`
Logical Or	Symbol:	||
	Description:	Binary operator which returns 1 if one or both of the expressions are nonzero, 0 otherwise
	Precedence:	4
	Example:	`ldi r18,Low(c1||c2) ;Load r18 with an expression`

Funções

The following functions are defined:

- LOW(expression) returns the low byte of an expression
- HIGH(expression) returns the second byte of an expression
- BYTE2(expression) is the same function as HIGH
- BYTE3(expression) returns the third byte of an expression
- BYTE4(expression) returns the fourth byte of an expression
- LWRD(expression) returns bits 0-15 of an expression
- HWRD(expression) returns bits 16-31 of an expression
- PAGE(expression) returns bits 16-21 of an expression
- EXP2(expression) returns 2^expression
- LOG2(expression) returns the integer part of log2(expression)