

**UNIVERSIDADE REGIONAL DE BLUMENAU**  
**CENTRO DE CIÊNCIAS EXATAS E NATURAIS**  
**CURSO DE CIÊNCIAS DA COMPUTAÇÃO**  
(Bacharelado)

**PROTÓTIPO DE AMBIENTE LADDER PARA OS  
MICROCONTROLADORES 8051 E PIC16F873**

TRABALHO DE CONCLUSÃO DE CURSO SUBMETIDO À UNIVERSIDADE  
REGIONAL DE BLUMENAU PARA A OBTENÇÃO DOS CRÉDITOS NA  
DISCIPLINA COM NOME EQUIVALENTE NO CURSO DE CIÊNCIAS DA  
COMPUTAÇÃO — BACHARELADO

**SIDNEI ALEXANDRE DE ALMEIDA**

BLUMENAU, JUNHO/2003

2003/2-23

# **PROTÓTIPO DE AMBIENTE LADDER PARA OS MICROCONTROLADORES 8051 E PIC16F873**

**SIDNEI ALEXANDRE DE ALMEIDA**

ESTE TRABALHO DE CONCLUSÃO DE CURSO, FOI JULGADO ADEQUADO  
PARA OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA DE TRABALHO DE  
CONCLUSÃO DE CURSO OBRIGATÓRIA PARA OBTENÇÃO DO TÍTULO DE:

**BACHAREL EM CIÊNCIAS DA COMPUTAÇÃO**

---

Prof. Antônio Carlos Tavares — Orientador na FURB

---

Prof. José Roque Voltolini da Silva — Coordenador do TCC

## **BANCA EXAMINADORA**

---

Prof. Antônio Carlos Tavares

---

Prof. Mauro Mattos

---

Prof. Miguel Alexandre Wisintainer

## RESUMO

Este trabalho apresenta a implementação de um ambiente de linguagem *Ladder*. A partir do diagrama, este ambiente é capaz de gerar código *assembly* para os microcontroladores 8051 e PIC16F873. Para implementação utilizaram-se técnicas de orientação a objeto e técnicas de construção de compiladores.

## **ABSTRACT**

This work describes the implementation of a language Ladder environment. Starting from a diagram, this tool is able to generate source code assembly to microcontrollers 8051 and PIC16F873. The software was built using object orientation techniques and compilers construction techniques.

## LISTA DE FIGURAS

FIGURA 1 -EXEMPLO DE UM DIAGRAMA LADDER .....	14
FIGURA 2 - PINAGEM DO MICROCONTROLADOR PIC16F873 .....	21
FIGURA 3 -DIAGRAMA DA ARQUITETURA INTERNA DO PIC16F873 .....	23
FIGURA 4 – FLUXOGRAMA DA CAPTURA A/D.....	30
FIGURA 5 - MAPA DA MEMÓRIA DO PIC16F873 .....	33
FIGURA 6 - PINAGEM DO 8051 .....	34
FIGURA 7 -MAPA DE MEMÓRIA DO 8051 .....	38
FIGURA 8 - DIAGRAMA DE UM COMPILADOR.....	39
FIGURA 9 - ÁRVORE DA OPERAÇÃO IF (A OR B) AND C THEN Z .....	40
FIGURA 10 – INTERFACE DO MPLAB.....	44
FIGURA 11 – TELA DE EDIÇÃO DO PROJETO.....	45
FIGURA 12 - DIAGRAMA DE CASOS DE USO .....	46
FIGURA 13 - DIAGRAMA DE CLASSES .....	47
FIGURA 14 - DIAGRAMA DA CLASSE SÍMBOLO .....	48
FIGURA 15 - DIAGRAMA DA CLASSE COMPILADOR.....	49
FIGURA 16 - DIAGRAMA DE SEQÜÊNCIA ESCOLHER MICROCONTROLADOR....	51
FIGURA 17 - DIAGRAMA DE SEQÜÊNCIA DESENHAR DIAGRAMA.....	52
FIGURA 18 - DIAGRAMA DE SEQÜÊNCIA GERAR CÓDIGO.....	53
FIGURA 19 - TELA DE ABERTURA DO PROTÓTIPO .....	56
FIGURA 20 - TELA DE ESCOLHA DO MICROCONTROLADOR.....	57
FIGURA 21 - CONFIGURAÇÃO DE PORTAS .....	58
FIGURA 22 - ESCOLHENDO A VARIÁVEL.....	59
FIGURA 23 - DIAGRAMA LADDER DE UM MEIO-SOMADOR .....	60
FIGURA 24 - CÓDIGO ASSEMBLY GERADO DO MEIO SOMADOR .....	61
FIGURA 25 – RESULTADO DA MONTAGEM .....	62

## LISTA DE TABELAS

TABELA 1 - PROPRIEDADES DO PIC16F873 .....	20
TABELA 2 - PINAGEM DO PIC16F873.....	21
TABELA 3 - REGISTRADORES DE FUNÇÃO ESPECIAL.....	24
TABELA 4 -REGISTRADORES ENVOLVIDOS NA CONFIGURAÇÃO DA PORTA .....	25
TABELA 5 -REGISTRADORES ENVOLVIDOS NA CONFIGURAÇÃO DA PORTB ....	26
TABELA 6 - REGISTRADORES ENVOLVIDOS NA CONFIGURAÇÃO DA PORTC ....	26
TABELA 7 - REGISTRADOR OPTION_REG .....	27
TABELA 8 – CONFIGURAÇÃO DA ORIGEM DO CLOCK.....	27
TABELA 9 – CONFIGURAÇÃO DO PRESCALER DO TMR0.....	28
TABELA 10 -REGISTRADOR T1CON .....	28
TABELA 11 – CONFIGURAÇÃO DA ORIGEM DO CLOCK DO TMR1 .....	29
TABELA 12 – CONFIGURAÇÃO DO PRESCALER DO TMR1 .....	29
TABELA 13 - REGISTRADORES PARA OPERAÇÃO DE CAPTURA A/D .....	31
TABELA 14 - REGISTRADOR DE STATUS.....	31
TABELA 15 – CONFIGURAÇÃO DOS BANCOS DE MEMÓRIA.....	31
TABELA 16 - REGISTRADORES DE FUNÇÃO ESPECIAL DO 8051 .....	36
TABELA 17 - REGISTRADORES USADOS NA CONFIGURAÇÃO DOS TEMPORIZADORES .....	37
TABELA 18 – CONFIGURAÇÃO DOS TEMPORIZADORES.....	37
TABELA 19 – ESQUEMA DE TRADUÇÃO PARA UMA LINGUAGEM DE TRÊS OPERANDOS .....	
TABELA 20 – ESQUEMA DE TRADUÇÃO PARA O PIC16F873 .....	54
TABELA 21 – ESQUEMA DE TRADUÇÃO PARA O 8051 .....	55

# SUMÁRIO

RESUMO .....	III
ABSTRACT .....	IV
LISTA DE FIGURAS .....	V
LISTA DE TABELAS .....	VI
1 INTRODUÇÃO .....	10
1.1 OBJETIVOS DO TRABALHO .....	11
1.2 ESTRUTURA DO TRABALHO .....	12
2 LINGUAGEM DIAGRAMA LADDER (LD) .....	13
2.1 LINHAS DE FORÇA .....	13
2.2 ELEMENTOS DE CONEXÃO E ESTADOS .....	13
2.3 CONTATOS .....	14
2.3.1 CONTATOS ESTÁTICOS .....	15
2.3.2 CONTATOS SENSORES DE TRANSIÇÃO .....	15
2.4 BOBINAS .....	15
2.4.1 BOBINAS MOMENTÂNEAS .....	16
2.4.2 BOBINAS <i>LATCH</i> .....	16
2.4.3 BOBINAS RETENTIVAS .....	16
2.4.4 BOBINAS SENSORES DE TRANSIÇÃO .....	17
2.5 FUNÇÕES E BLOCO DE FUNÇÕES .....	17
2.5.1 FUNÇÕES DE TEMPORIZADORES .....	17
2.5.2 FUNÇÕES DE CONTADORES .....	18
2.5.3 FUNÇÃO CONVERSOR ANALÓGICO/DIGITAL .....	18
2.6 ORDEM DE AVALIAÇÃO DO DIAGRAMA .....	19

3	MICROCONTROLADORES .....	20
3.1	MICROCONTROLADOR PIC16F873 .....	20
3.1.1	<i>SPECIAL FUNCTION REGISTERS</i> – SFR .....	23
3.1.2	CONJUNTO DE INSTRUÇÕES.....	24
3.1.3	PORTAS DE ENTRADA/SAÍDA.....	25
3.1.3.1	Registrador PORTA.....	25
3.1.3.2	Registrador PORTB.....	25
3.1.3.3	Utilizando a PORTC.....	26
3.1.4	TEMPORIZADORES.....	27
3.1.4.1	Temporizador TMR0.....	27
3.1.4.2	Temporizador TMR1 .....	28
3.1.5	CONVERSOR ANALÓGICO/DIGITAL (A/D).....	29
3.1.6	UTILIZANDO A MEMÓRIA .....	31
3.2	MICROCONTROLADOR 8051.....	34
3.2.1	SFR’S (REGISTRADORES ESPECIAIS).....	35
3.2.2	PORTAS DE ENTRADA E SAÍDA .....	36
3.2.3	TEMPORIZADORES.....	36
3.2.4	UTILIZANDO MEMÓRIA .....	37
4	COMPILADORES .....	39
4.1	MODELO DE COMPILAÇÃO DE ANÁLISE E SÍNTESE .....	39
4.2	BNF OU GRAMÁTICA LIVRE DE CONTEXTO.....	40
4.3	TRADUÇÃO DIRIGIDA PELA SINTAXE .....	41
4.4	GRAMÁTICA PARA EXPRESSÕES LÓGICAS .....	42
5	MPLAB.....	44
6	DESENVOLVIMENTO DO TRABALHO .....	46



6.1 ESPECIFICAÇÃO .....	46
6.2 ESQUEMAS DE TRADUÇÕES .....	54
6.3 OPERACIONALIDADE DA IMPLEMENTAÇÃO .....	55
6.3.1 PRINCIPAIS FUNÇÕES.....	55
6.3.2 INICIANDO UM NOVO PROJETO .....	57
6.3.3 CONFIGURANDO AS VARIÁVEIS .....	57
6.3.4 DESENHANDO O DIAGRAMA.....	59
6.3.5 COMPILANDO O DIAGRAMA .....	59
6.3.6 MONTANDO O CÓDIGO GERADO .....	61
7 CONCLUSÕES .....	63
7.1 EXTENSÕES .....	63
REFERÊNCIAS BIBLIOGRÁFICAS .....	64
ANEXO 1 – CONJUNTO DE INSTRUÇÕES DO PIC16F873.....	65
ANEXO 2 – CONJUNTO DE INSTRUÇÕES DO 8051 .....	66
ANEXO 3 – ARQUIVO DE CONFIGURAÇÃO DO PIC16F873 .....	69
ANEXO 4 – ARQUIVO DE CONFIGURAÇÃO DO 8051 .....	74

# 1 INTRODUÇÃO

Grandes empresas que desenvolvem circuitos integrados estão freqüentemente lançando no mercado novos microcontroladores com recursos computacionais já inclusos no encapsulamento mais conhecidos como *embedded* (programas embutidos).

Todo microcontrolador dispõe de uma linguagem de programação para que se possa programá-lo. Esta linguagem é o *assembly*, linguagem em que a abstração dos dados é muito menor do que aquela apresentada nas linguagens de alto nível.

Há um grande interesse, por parte dos alunos da computação em aprender sobre programação de microcontroladores, mas a grande barreira está na complexidade da programação envolvida nos mesmos, muito diferente das linguagens de alto nível.

Outra motivação para este interesse, é que os microcontroladores têm muitas características em comum com um *personal computer* (PC), como por exemplo, portas seriais, portas paralelas etc; além de serem compactos, discretos e de baixo custo.

Apesar de existirem ferramentas que possibilitam a utilização de linguagens conhecidas como C, Pascal ou Basic, o preço destas ferramentas está além das possibilidades dos alunos.

Uma linguagem de programação genérica para microcontroladores evitaria o esforço de ter que aprender as várias linguagens existentes para cada família destes.

Para tal fim foram criadas linguagens orientadas a programação de microcontroladores.

Uma destas linguagens, conhecida como *Ladder*, é baseada na simbologia gráfica e seus arranjos. Esta linguagem representa o fluxo da corrente elétrica. Possui símbolos que representam contatos (*ON/OFF* - entradas que representam dados do mundo real) e símbolos que representam o comportamento de uma saída quanto à mudança das entradas. Outros símbolos (na forma de retângulos) representam uma macro, ou seja, um conjunto de operações.

Sendo assim, um software capaz de construir e interpretar esta simbologia e convertê-la em código *assembly* para o microcontrolador escolhido seria interessante. O contato com a linguagem *Ladder* possibilita também que o aluno familiarize-se com a linguagem muito utilizada usado em controladores lógicos programáveis (CLP's).

Este trabalho é uma continuação do trabalho intitulado “Ambiente *Ladder* para o PIC16F874 e 8051” apoiado pelo programa PIPE da Fundação Universidade Regional de Blumenau, realizado no ano de 2000, desenvolvido pelo autor deste trabalho. Entre os resultados obtidos com o referido projeto PIPE, obteve-se uma ferramenta para geração de código *assembly* para o microcontrolador PIC16F84, sem haver a necessidade de conhecimento da linguagem *assembly* do mesmo. Porém, com a inclusão de novos microcontroladores e novas funções, aumenta-se a complexidade do sistema e torna-se necessário rever as técnicas e modelagem empregadas naquele sistema.

Utilizando técnicas de programação existentes para construção de compiladores e modelagem orientada a objetos através da UML, propõe-se refazer o editor que permita representar o diagrama *Ladder*, e a partir deste diagrama a conversão para a linguagem *assembly* específica do respectivo microcontrolador.

## 1.1 OBJETIVOS DO TRABALHO

O objetivo do trabalho é desenvolver uma ferramenta de geração de código *assembly* para os microcontroladores 8051 e PIC16F873, a partir da representação em diagramas *Ladder*.

Os objetivos específicos deste trabalho são:

- a) especificar orientado a objeto, utilizando UML e implementar o código a partir desta especificação;
- b) especificar a linguagem dos microcontroladores usando a forma de Backus-Naur (BNF);
- c) inclusão dos microcontroladores 8051 e PIC16F873 (atualmente só trabalha com o PIC16F84);
- d) inclusão dos blocos de função que permita comparações de sinais analógicos, uso de temporizadores e uso de contadores.
- e) documentar a linguagem *Ladder*.

## 1.2 ESTRUTURA DO TRABALHO

O primeiro capítulo apresenta os motivos que levaram o autor à realização deste trabalho.

A teoria relativa à linguagem *Ladder* e todos os símbolos utilizados serão apresentados no capítulo 2.

No capítulo 3 será apresentada uma pequena introdução aos microcontroladores. No tópico 3.1 será apresentado o microcontrolador PIC16F873 e suas operações básicas de funcionamento. Apresentará como configurar os elementos necessários ao projeto e apresentará também como está organizada a memória deste componente.

No tópico 3.2 será apresentado o microcontrolador 8051 e suas funcionalidades. Também será apresentada a configuração dos elementos e como está organizada a memória deste componente.

No capítulo 4 será apresentada a teoria básica de compiladores que foi utilizado neste projeto.

Uma breve explicação sobre o montador MPLab será apresentado no capítulo 5.

A especificação, utilizando UML, e a análise de requisitos estarão detalhadas no capítulo 6. Neste capítulo também estará especificada a linguagem dos microcontroladores em BNF.

O funcionamento através de um estudo de caso será apresentado no capítulo 7.

## 2 LINGUAGEM DIAGRAMA LADDER (LD )

Um programa LD habilita o controlador programável a testar e modificar dados por meio de símbolos gráficos padronizados. Estes símbolos estão dispostos em redes de maneira similar a degraus de uma escada de relés na forma de um diagrama lógico. O diagrama *ladder* é limitado na esquerda e na direita por linhas de força. Na figura 1 um exemplo de um diagrama *ladder*.

### 2.1 LINHAS DE FORÇA

O diagrama *Ladder* deve ser delimitado na esquerda e na direita por linhas verticais chamadas de linha de força da esquerda e de linha de força da direita, respectivamente. A linha de força da direita pode ser explícita ou implícita.

### 2.2 ELEMENTOS DE CONEXÃO E ESTADOS

Como exibidos na figura 1, elementos de conexão podem ser horizontais ou verticais. Os estados dos elementos de conexão devem ser denotados em ligado ou desligado, correspondendo a um valor literal lógico 1 ou 0, respectivamente. O estado da conexão deve ser sinônimo ao termo fluxo de energia.

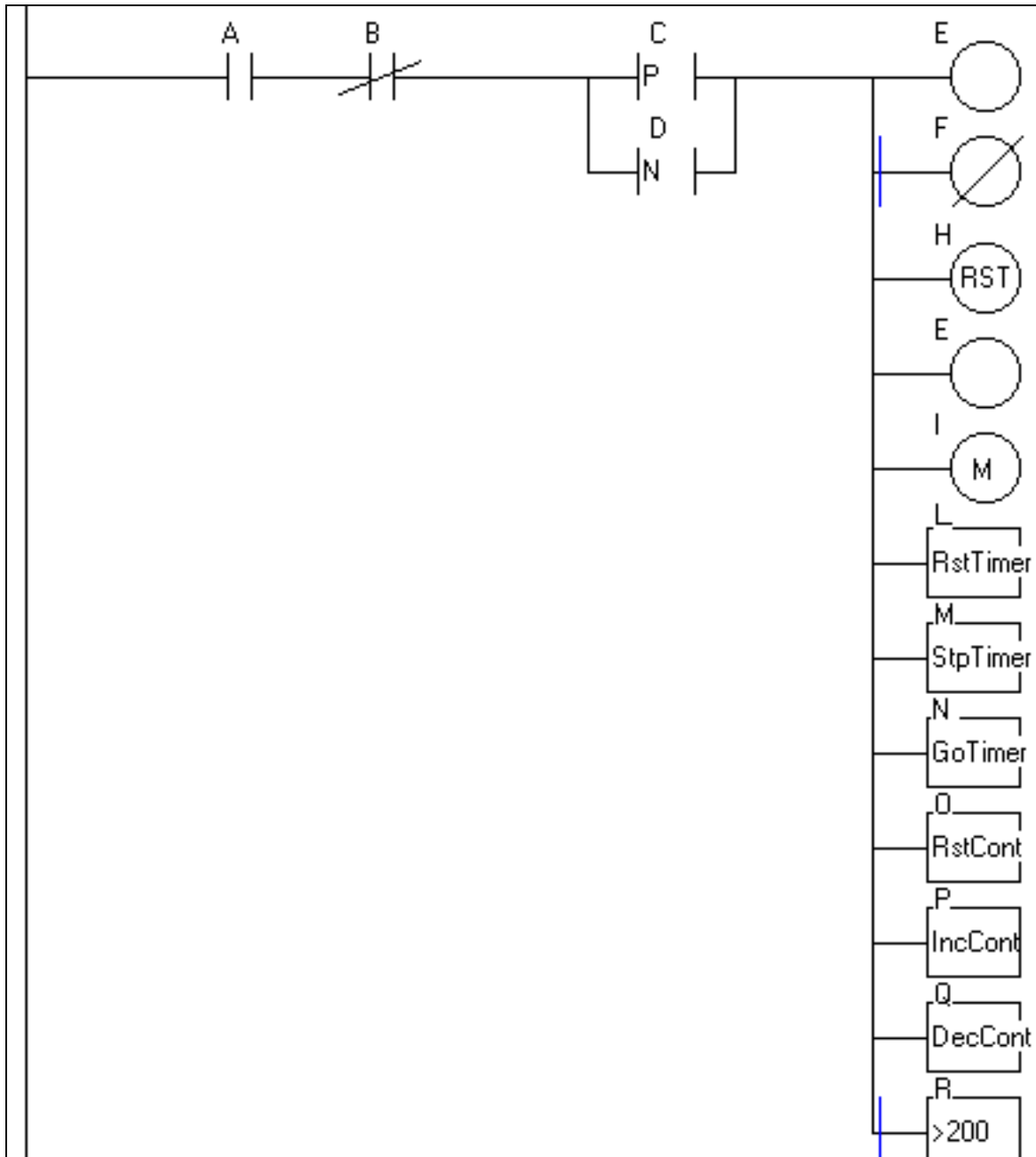
O estado da linha de força da esquerda deve sempre ser considerado Ligado ou 1. Nenhum estado é definido para a linha da direita.

Um elemento de conexão horizontal deve ser indicado por uma linha horizontal. Um elemento de conexão horizontal transmite o estado do elemento da esquerda imediata para o elemento em sua direita imediata.

Um elemento de conexão vertical deve consistir de uma linha vertical interseccionando-se com uma ou mais conexões horizontais em cada lado. O estado de uma conexão vertical deve representar o OU inclusivo dos estados ligados de uma conexão horizontal em sua esquerda, isto é, o estado de uma conexão vertical deve ser desligado se os estados de todas as conexões horizontais reunidas à esquerda estão desligados; ligado se o estado de uma ou mais das conexões horizontais reunidas à esquerda estão ligadas. Na figura 1, a conexão vertical liga os contatos C e D.

O estado de uma conexão horizontal deve ser copiado para todas as conexões horizontais reunidas a direita.

FIGURA 1 -EXEMPLO DE UM DIAGRAMA LADDER



## 2.3 CONTATOS

Um contato é um elemento que concede um estado a uma conexão horizontal a sua direita que é igual a uma operação lógica E do estado da conexão horizontal ao seu lado esquerdo com uma função apropriada a uma variável lógica associada de entrada, de saída ou

de memória. O contato não modifica o valor da variável lógica associada ao símbolo da esquerda.

### **2.3.1 CONTATOS ESTÁTICOS**

Os contatos estáticos se dividem em contatos normalmente abertos e normalmente fechados.

Em um contato normalmente aberto, o estado da conexão da esquerda é copiado para conexão da direita se o estado da variável lógica associado está ligado. Em outros casos o estado da conexão da direita é desligado. Na figura 1, o símbolo A representa um contato normalmente aberto.

Em um contato normalmente fechado, o estado da conexão da esquerda é copiado para a conexão da direita se o estado da variável lógica associado está desligado. Em outros casos o estado da conexão da direita será desligado. Na figura 1, o símbolo B representa um contato normalmente fechado.

### **2.3.2 CONTATOS SENSORES DE TRANSIÇÃO**

Os contatos de sensores de transição podem ser classificados em sensores de transição positiva e sensores de transição negativa.

Em um contato sensor de transição positiva, o estado do contato será ligado se for percebida uma transição de desligado para ligado ou de 0 para 1 da avaliação dos símbolos à esquerda deste. O estado do contato será desligado toda as outras vezes. O contato sensor de transição positiva está representado pelo símbolo C na Figura 1.

Em um contato sensor de transição negativa, O estado do contato será ligado ou 1 se for percebida uma transição de ligado para desligado ou de 1 para 0 da avaliação dos símbolos à esquerda deste. O estado do contato será desligado toda as outras vezes. O contato sensor de transição negativa está representado pelo símbolo D na Figura 1.

## **2.4 BOBINAS**

Uma bobina copia o estado dos contatos a sua esquerda para a conexão a sua direita sem modificação, e armazena uma função apropriada do estado ou transição da conexão da esquerda em uma variável lógica associada.

As bobinas podem ser divididas em momentâneas, latch, sensoras de transição e retentivas.

### 2.4.1 BOBINAS MOMENTÂNEAS

As bobinas momentâneas podem ser classificadas em normal e negada.

Em uma bobina normal, o estado da conexão da esquerda é copiado para a variável lógica associada e para a conexão da direita. Uma bobina está representada na Figura 1, pelo símbolo E.

Em uma bobina negada, o estado da conexão da esquerda é copiado para a conexão da direita. O inverso do estado da conexão da esquerda é copiado para a variável lógica associada, isto é, se o estado da conexão da esquerda for desligado, então o estado da variável associada será ligado e vice-versa. Uma bobina negada está representada na Figura 1, pelo símbolo F.

### 2.4.2 BOBINAS LATCH

As bobinas *latch* podem ser divididas em *set* e *reset*.

Em uma bobina *SET (latch)*, a variável lógica associada está ativada para o estado ligado quando a conexão da esquerda está ligada, e permanecerá ativada até ser desativada por uma bobina *RESET*. A bobina *SET* está representada no diagrama da Figura 1, pelo símbolo G.

Em uma bobina *RESET (unlatch)*, a variável lógica associada é desativada para o estado desligado quando a conexão da esquerda está no estado ligado, e permanecerá desativada até ser ativada por uma bobina *SET*. A bobina *RESET* está representada no diagrama da Figura 1, pelo símbolo H.

### 2.4.3 BOBINAS RETENTIVAS

As bobinas retentivas seguem o mesmo comportamento das bobinas normais e bobinas *latch* já descritas anteriormente. A diferença entre as bobinas retentivas e as bobinas normais é que as bobinas retentivas representam memórias utilizadas para auxiliar a confecção de programas na linguagem *Ladder*, enquanto as bobinas normais representam portas de entrada e saída.



Bobinas retentivas têm a mesma representação das bobinas normais, porém dentro do círculo é adicionado um M, conforme representado na Figura 1, pelo símbolo I.

#### **2.4.4 BOBINAS SENSORES DE TRANSIÇÃO**

As bobinas sensoras de transição podem ser divididas em sensoras de transição positiva e negativa.

Em uma bobina sensora de transição positiva, o estado da variável lógica associada está ligado a partir de uma avaliação deste elemento para o próximo quando a transição da conexão da esquerda for de desligado para ligado, ou de 0 para 1. O estado da bobina será ligado ou 1, quando este evento acontecer. A bobina sensora de transição positiva está representada na Figura 1, pelo símbolo J.

Em uma bobina sensora de transição negativa, o estado da variável lógica associada será ligado a partir de uma avaliação deste elemento para o próximo quando a transição da conexão da esquerda for de ligado para desligado, ou de 1 para 0. O estado da bobina será ligado ou 1, quando este evento acontecer. A bobina sensora de transição negativa está representada na Figura 1, pelo símbolo K.

### **2.5 FUNÇÕES E BLOCO DE FUNÇÕES**

A representação de funções e blocos de funções na linguagem Ladder é definida através da figura de um retângulo. Desde que a função possua no mínimo uma entrada e uma saída lógica.

Este projeto utiliza as seguintes funções abaixo.

#### **2.5.1 FUNÇÕES DE TEMPORIZADORES**

Os temporizadores são previamente configurados em um valor qualquer de tempo, e quando atingem o mesmo, provocam uma mudança de estado lógico no contato correspondente. As seguintes funções são usadas para controlar os temporizadores.

A função reseta *timer* reseta o temporizador, ou seja, reinicia a contagem do tempo a partir do zero. A função está representada no diagrama da Figura 1, pelo símbolo L.

A função pára *timer* interrompe a ação do temporizador mantendo a contagem do tempo até o instante da interrupção. A função está representada no diagrama da Figura 1, pelo símbolo M.

A função avança timer retoma a contagem do temporizador após uma interrupção. A função está representada no diagrama da Figura 1, pelo símbolo N.

## 2.5.2 FUNÇÕES DE CONTADORES

Os contadores, da mesma forma que os temporizadores, são configurados previamente. Quando atingem o valor pré-estabelecido, provocam uma mudança de estado no contato correspondente.

As seguintes funções definem o controle das funções dos contadores.

A função reseta contador reinicia a contagem, zerando a variável associada ao contador. A função está representada no diagrama da Figura 1, pelo símbolo O.

A função incrementa contador adiciona uma unidade ao valor prévio da variável associada ao contador. A função está representada no diagrama da Figura 1, pelo símbolo P.

A função decrementa contador subtrai uma unidade do valor prévio da variável associada ao contador. A função está representada no diagrama da Figura 1, pelo símbolo Q.

## 2.5.3 FUNÇÃO CONVERSOR ANALÓGICO/DIGITAL

A função compara A/D compara o valor obtido da porta conversora de analógico para digital (caso o microcontrolador a possua), com um valor previamente definido. A comparação pode ser maior, menor ou igual ao valor previamente definido.

Quando o valor analógico, comparado ao valor previamente configurado permitir que a expressão lógica configurada seja verdadeira, ativará o *flag* associado ao símbolo. Se for falsa desativará o *flag*.

A função está representada no diagrama da Figura 1, pelo símbolo R.

## **2.6 ORDEM DE AVALIAÇÃO DO DIAGRAMA**

A ordem de avaliação dos símbolos do diagrama Ladder é importante para determinar os valores das saídas e ativação das funções. Através desta ordem que se estabelece o fluxo de controle de operações lógicas no diagrama.

A ordem de avaliação dos símbolos do diagrama Ladder é sequencial e deve ser obedecer à ordem do topo para baixo conforme eles aparecem no diagrama. Ou seja avaliam-se primeiros os circuitos mais acima e por último os que estão mais abaixo no diagrama.

### 3 MICROCONTROLADORES

Microcontrolador é um dispositivo que agrega todos os componentes necessários a automação de sistemas como portas de entrada e saída, temporizadores, contadores etc. O microcontrolador corresponde a um microprocessador e seus periféricos típicos, todos juntos num só chip (NICOLSI, 2001). Microcontrolador (anteriormente chamado de microcomputador de um só *chip*), é um componente que possui todos os periféricos comuns embutidos em uma só pastilha, facilitando assim o desenvolvimento de sistemas pequenos e baratos, embora complexos e sofisticados (SILVA, 1998).

#### 3.1 MICROCONTROLADOR PIC16F873

Este microcontrolador fabricado pela Microchip© possui 28 pinos, sendo 22 de portas I/O, foi escolhido por possuir além de todos elementos habituais encontrados nos microcontroladores, como temporizadores, memória, portas de entrada e saída, apresenta conversor de analógico para digital (A/D) de 10 bits. O PIC16F873 possui arquitetura Harvard, onde barramentos de memória e instruções estão separados. Este microcontrolador também possui um reduzido número de instruções para ganhos de performance no processamento de instruções.

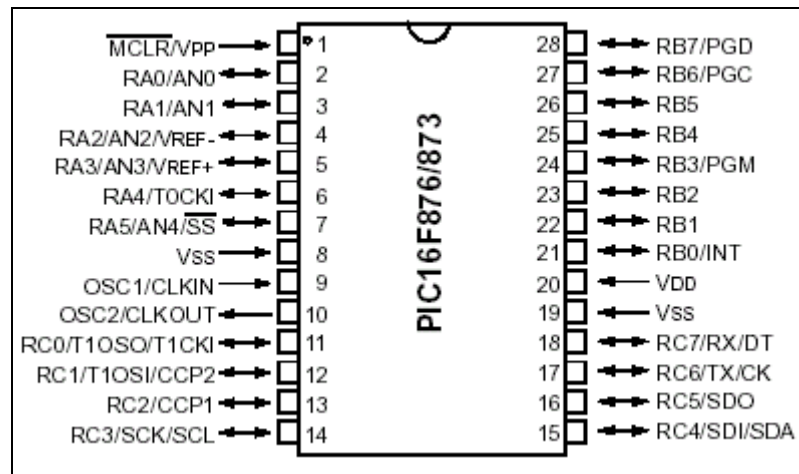
Segundo Microchip (2003), o microcontrolador possui as seguintes configurações demonstradas na Tabela 1.

**TABELA 1 - PROPRIEDADES DO PIC16F873**

Frequência de operação	DC 20 MHz
Memória de programa	4k (palavras de 14 bits)
Memória de dados	192 <i>bytes</i>
Memória de dados EEPROM (eletricamente programável)	128 <i>bytes</i>
Interrupções	13
Portas de entrada e saída	Portas A,B, C
Temporizadores	3
Modo A/D (analógico para digital) de 10 bits	5 canais de entrada
Conjunto de instruções	35

A figura 2, apresenta o microcontrolador e sua pinagem.

FIGURA 2 - PINAGEM DO MICROCONTROLADOR PIC16F873



A tabela 2, descreve detalhadamente as funções dos pinos do PIC16F873.

TABELA 2 - PINAGEM DO PIC16F873

OSC1/CLKIN	Entrada para o oscilador de cristal
OSC2/CLKOUT	Saída para o oscilador de cristal
MCLR/VPP	Reinicia o microcontrolador, zerando as memórias e registradores
RA0/AN0	Porta de entrada ou saída, pode também ser A/D
RA1/AN1	Porta de entrada ou saída, pode também ser A/D
RA2/AN2/VREF-	Porta de entrada ou saída, pode também ser A/D, ou tensão de referência negativa.
RA3/AN3/VREF+	Porta de entrada ou saída, pode também ser A/D ou tensão de referência positiva.
RA4/T0CKI	Porta de entrada ou saída, pode também ser entrada para o <i>Timer0</i>
RA5/SS/AN4	Porta de entrada ou saída, pode também ser A/D
RB0/INT	Porta de entrada ou saída, pode ser também pino de interrupção externa.
RB1	Porta de entrada e saída
RB2	Porta de entrada e saída
RB3/PGM	Porta de entrada e saída, pode ser também a entrada de baixa voltagem programada
RB4	Porta de entrada e saída, pode ser também sensor de transição
RB5	Porta de entrada e saída, pode ser também sensor de transição
RB6/PGC	Porta de entrada e saída, pode ser também sensor de transição.
RB7/PGD	Porta de entrada e saída, pode ser também sensor de transição
RC0/T1OSO/T1CKIO	Porta de entrada e saída, pode ser a saída para o oscilador do <i>Timer1</i> ou <i>clock</i> de entrada para o <i>Timer1</i>

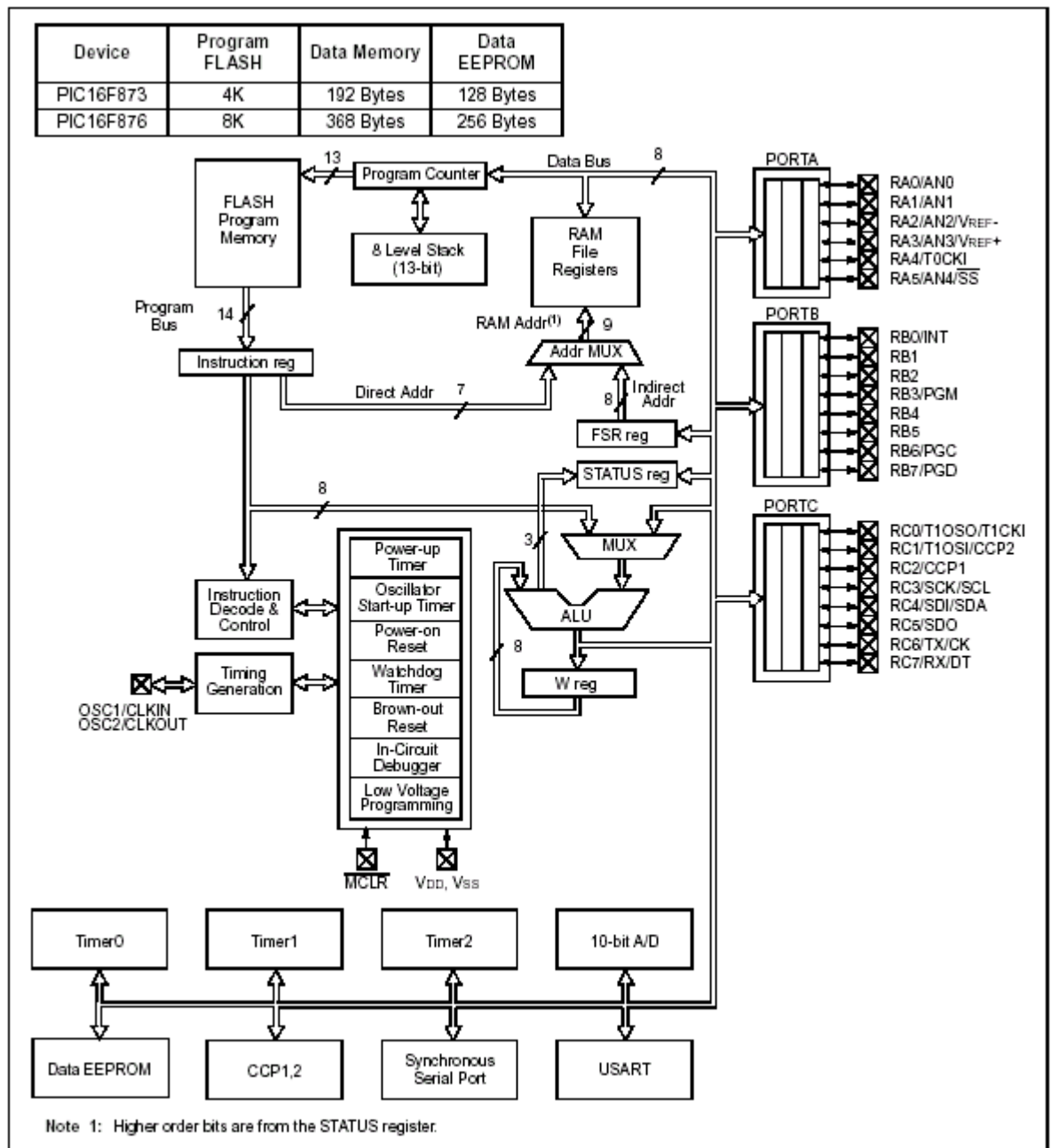
## CONTINUAÇÃO DA TABELA 2

RC1/T1OSI/CCP2	Porta de entrada ou saída, pode ser a saída para o oscilador do Timer1 ou <i>clock</i> de entrada para o <i>Timer1</i> ou <i>Capture2</i> entrada/ <i>Compare2</i> saída/PWM2 saída.
RC2/CCP1	Porta de entrada ou saída, pode ser também <i>Capture1</i> entrada/ <i>Compare1</i> saída/PWM1 saída.
RC3/SCK/SCL	Porta de entrada ou saída, pode ser também o <i>clock</i> serial síncrono entrada/saída para ambos SPI e I2C modos
RC4/SDI/DAS	Porta de entrada ou saída, pode ser também SPI <i>Data In</i> (SPI modo) ou dados I/O (I2C modo).
RC5/SDO	Porta de entrada ou saída, pode ser também SPI <i>Data Out</i> (SPI modo).
RC6/TX/CK	Porta de entrada ou saída, pode ser também a USART transmissor assíncrono ou <i>clock</i> síncrono.
RC7/RX/DT	Porta de entrada ou saída, pode ser também a USART receptor assíncrono ou dados síncronos.
VSS	Referência de terra para lógica e pinos I/O
VDD	Referência positiva para lógica e pinos I/O.

É um microcontrolador de arquitetura Harvard, que prevê várias vias de comunicação entre CPU e periféricos, permitindo a realização de várias operações simultaneamente, o que implica em aumento considerável na velocidade de execução e permite ainda que as memórias de dados e a de programa tenham tamanhos diferentes (SILVA, 1998).

A arquitetura deste microcontrolador está ilustrada na Figura 3.

FIGURA 3 -DIAGRAMA DA ARQUITETURA INTERNA DO PIC16F873



### 3.1.1 SPECIAL FUNCTION REGISTERS – SFR

Os SFR's (registradores de funções especiais) são muito utilizados na configuração dos elementos que compõem o microcontrolador. Servem para configurar portas, ativar interrupções e outras funções. Segundo Microchip, 2003 os mais usados registradores de funções especiais do PIC16F873 são os seguintes representados na Tabela 3.

TABELA 3 - REGISTRADORES DE FUNÇÃO ESPECIAL

Endereço	Registrador	Função
Bank0		
01h	TMR0	Módulo do <i>Timer0</i>
05h	PORTA	Registrador de estado da Porta A
06h	PORTB	Registrador de estado da Porta B
07h	PORTC	Registrador de estado da Porta C
0Eh	TMR1L	Módulo dos bits menos significativos do <i>Timer1</i>
0Fh	TMR1H	Módulo dos bits mais significativos do <i>Timer1</i>
10h	T1CON	Controle do <i>Timer1</i>
1Eh	ADRESH	Módulo dos bits mais significativos da captura A/D
1Fh	ADCON0	Controle da captura A/D
Bank1		
81h	OPTION_REG	Configuração do registrador <i>Timer0</i>
85h	TRISA	Configuração do modo de operação da Porta A
86h	TRISB	Configuração do modo de operação da Porta B
87h	TRISC	Configuração do modo de operação da Porta C
8Bh	INTCON	Controle de interrupções
9Eh	ADRESL	Módulo dos bits menos significativos da captura A/D
9Fh	ADCON1	Controle da captura A/D

Estes registradores serão explicados de forma mais detalhada nos itens a seguir.

### 3.1.2 CONJUNTO DE INSTRUÇÕES

O PIC está encaixado na categoria RISC (conjunto de instruções reduzido), por possuir apenas 35 instruções que atendem ao funcionamento básico de um sistema microcontrolado. Como os barramentos de memória e de programa estão separados o tamanho da palavra de memória das instruções não precisa ter o mesmo tamanho da memória de dados. No PIC o tamanho do código de operação é de 14 bits. Isto permite que instruções que levariam 2 passos ou mais sejam executados em uma só passagem.

No anexo 1 estão relacionadas as instruções do microcontrolador PIC16F873.



### 3.1.3 PORTAS DE ENTRADA/SAÍDA

As portas são o elo de ligação do microcontrolador com o mundo exterior. Através delas que se realizam todas as comunicações com os periféricos. O PIC16F873 possui 22 portas com diversas funcionalidades.

Estas funcionalidades serão apresentadas a seguir.

#### 3.1.3.1 REGISTRADOR PORTA

Segundo Microchip (2003), PORTA é uma porta bidirecional (entrada ou saída) de 6-bit de largura. O registrador correspondente a configuração da direção dos dados (saída ou entrada) é o registrador TRISA. Configurando o bit TRISA igual a 1 torna o correspondente bit do registrador PORTA uma entrada e configurando como 0 uma saída.

Os pinos de PORTA também são multiplexados com entradas analógicas. A operação de cada pino é selecionada pela configuração dos bits de controle do registrador ADCON1. O usuário também deve assegurar que o bit correspondente no registrador TRISA da porta usada esteja configurado em 1 quando usar entradas analógicas.

A Tabela 4 apresenta os registradores usados para configuração da PORTA e seus respectivos pinos.

**TABELA 4 -REGISTRADORES ENVOLVIDOS NA CONFIGURAÇÃO DA PORTA**

Endereço	Nome	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
05h	PORTA			RA5	RA4	RA3	RA2	RA1	RA0
85h	TRISA	Registrador de direção de dados da PORTA							
9Fh	ADCON1					PCFG3	PCFG2	PCFG1	PCFG0

#### 3.1.3.2 REGISTRADOR PORTB

Segundo Microchip (2003), PORTB é uma porta bidirecional de 8-bits de largura. O registrador correspondente a configuração da direção dos dados (saída ou entrada) é o

registrador TRISB. Configurando o bit TRISB igual a 1 torna o correspondente bit do registrador PORTB uma entrada e configurando como 0 uma saída.

Os pinos de PORTB podem ser programados sensores de transição. Para tal operação os pinos do TRISB devem estar configurados como entradas. O pino RBIF do registrador INTCON também deve estar configurado em 0.

A Tabela 5 apresenta os registradores usados para configuração da PORTA e seus respectivos pinos.

**TABELA 5 -REGISTRADORES ENVOLVIDOS NA CONFIGURAÇÃO DA PORTB**

Endereço	Nome	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
06h, 106h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
86h, 186h	TRISB	Registrador de direção de dados da PORTB							
8Bh	INTCON								RBIF

### 3.1.3.3 UTILIZANDO A PORTC

Segundo Microchip (2003), PORTC é uma porta bidirecional de 8-bits de largura. O registrador correspondente à configuração da direção dos dados (saída ou entrada) é o registrador TRISC. Configurando o bit TRISC igual a 1 torna o correspondente bit do registrador PORTC uma entrada e configurando como 0 uma saída.

A Tabela 6 apresenta os registradores usados para configuração da PORTA e seus respectivos pinos.

**TABELA 6 - REGISTRADORES ENVOLVIDOS NA CONFIGURAÇÃO DA PORTB**

Endereço	Nome	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
07h	PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0
87h	TRISC	Registrador de direção de dados da PORTC							

### 3.1.4 TEMPORIZADORES

Os temporizadores são utilizados quando se deseja ter controle sobre o tempo de alguma determinada tarefa. O PIC16F873 possui 3 temporizadores TMR0 (Timer 0), TMR1 (Timer 1) e TMR2 (Timer 2). Os temporizadores do PIC16F873 podem ser configurados por clock interno ou externo, exceto o TMR2, que só pode ser configurado como externo.

O funcionamento e os registradores envolvidos na configuração deste registradores serão detalhado a seguir.

#### 3.1.4.1 TEMPORIZADOR TMR0

O TMR0 ou temporizador 0 é um registrador de 8 bits que é incrementado automaticamente conforme as configurações dos registradores relacionados. Este registrador trabalha com a frequência do oscilador, ou seja, 20 MHz. Este temporizador não tem controle de avanço e parada.

Para selecionar a origem do clock este recurso é necessário configurar o registrador OPTION\_REG (registrador de opções). Este registrador permite configurar o comportamento do TMR0. A tabela 7 apresenta este registrador e os bits envolvidos na configuração do TMR0. A tabela 8 demonstra como configurar a origem do clock e o para quem se destina o prescaler através dos bits de controle.

**TABELA 7 - REGISTRADOR OPTION\_REG**

Endereço	Nome	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
81H	OPTION_REG			TOCS	TOSE	PSA	PS2	PS1	PS0

**TABELA 8 - CONFIGURAÇÃO DA ORIGEM DO CLOCK**

<b>bit 5 TOCS(Timer 0 clock source) TMR0 seletor da origem do Clock bit</b>	
1	Transição no pino TOCKI
0	Clock Interno
<b>bit 3 PSA (Prescaler assignment) Configuração do Prescaler</b>	
1	Prescaler é atribuído ao WDT ( <i>watchdog timer</i> )
0	Prescaler é atribuído ao módulo do Timer0

Este temporizador possui configuração de escala através dos bits 0 ao bit 4 do registrador OPTION\_REG. A escala serve para determinar tempos curtos ou tempos longos. A tabela 9 demonstra quais são os bits e as opções de escalas relacionadas para o registrador TMR0 e como configurá-las.

**TABELA 9 - CONFIGURAÇÃO DO PRESCALER DO TMR0**

Bit			bits de seleção do fator do <i>Prescaler</i>
PS2	PS1	PS0	
0	0	0	1:2 (incrementa Timer0 a cada dois ciclos do <i>clock</i> )
0	0	1	1:4 (incrementa Timer0 a cada 4 ciclos do <i>clock</i> )
0	1	0	1:8 (incrementa Timer0 a cada 8 ciclos do <i>clock</i> )
0	1	1	1:16 (incrementa Timer0 a cada 16 ciclos do <i>clock</i> )
1	0	0	1:32 (incrementa Timer0 a cada 32 ciclos do <i>clock</i> )
1	0	1	1:64 (incrementa Timer0 a cada 64 ciclos do <i>clock</i> )
1	1	0	1:128 (incrementa Timer0 a cada 128 ciclos do <i>clock</i> )
1	1	1	1:256 (incrementa Timer0 a cada 256 ciclos do <i>clock</i> )

### 3.1.4.2 TEMPORIZADOR TMR1

O TMR1L e o TMR1H juntos formam um registrador de 16 bits para temporizador 1 que é incrementado automaticamente conforme as configurações dos registradores relacionados. Este registrador trabalha com a frequência do oscilador dividida por 4, ou seja para um *clock* de 20MHZ sua frequência será de 5 MHZ.

O comportamento do temporizador TMR1 é determinado pelo T1CON (Controle do do Timer1). O registrador T1CON e seus bits de controle estão demonstrados na tabela 10.

**TABELA 10 -REGISTRADOR T1CON**

Endereço	Nome	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
10H	T1CON			T1CKPS1	T1CKPS0			TMR1CS	TMR1ON

Para configurar a origem do clock do TMR1 deve-se configurar o bit TMR1CS (origem do clock do Timer1) conforme demonstrado na tabela 11.

TABELA 11 - CONFIGURAÇÃO DA ORIGEM DO CLOCK DO TMR1

bit 1	TMR1CS: bit de seleção da origem do clock do Timer1
1	<i>Clock</i> externo do pino RC0/T1OSO/T1CKI
0	<i>Clock</i> Interno (Frequência do oscilador/4)

Este temporizador também possui controle de escala que são determinados pelos bits 5 e 4 do registrador T1CON. As opções de escalas estão apresentadas na tabela 12.

TABELA 12 - CONFIGURAÇÃO DO PRESCALER DO TMR1

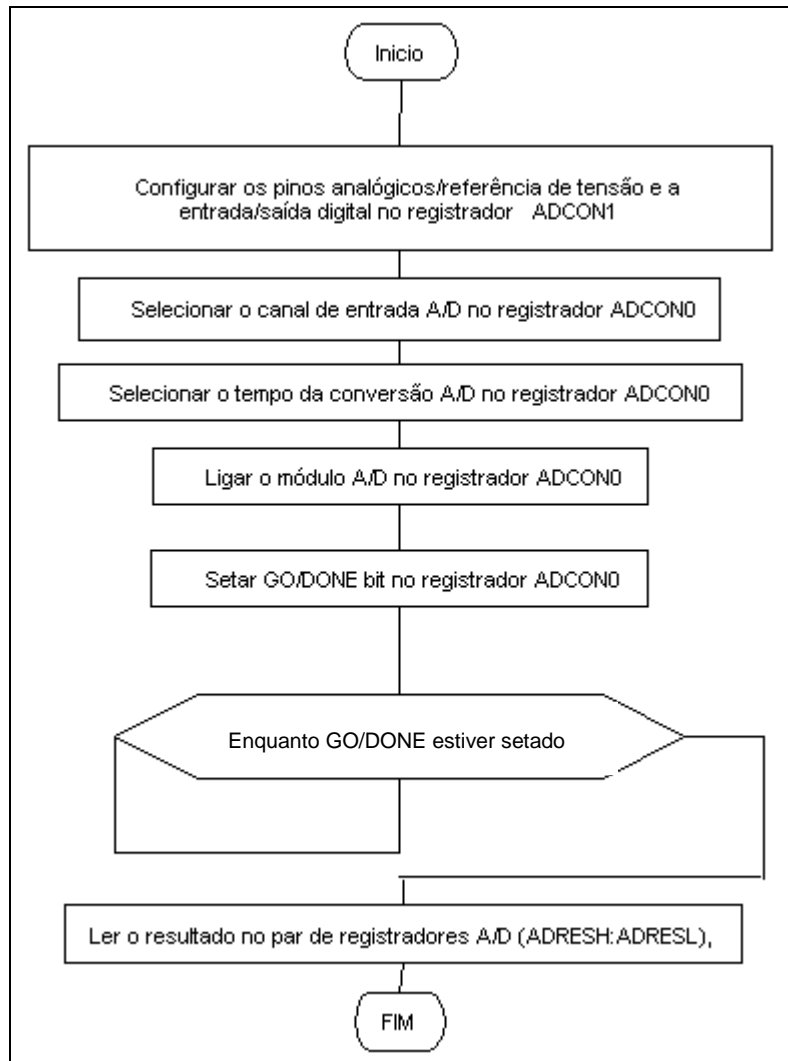
T1CKPS1	T1CKPS0	bits de seleção do prescaler do Timer1	
1	1	1:8	incremento dos registradores a cada 8 ciclos do clock
1	0	1:4	incremento dos registradores a cada 4 ciclos do clock
0	1	1:2	incremento dos registradores a cada 2 ciclos do clock
0	0	1:1	incremento dos registradores a cada 1 ciclo do clock

O TMR1 diferente do TMR0, possui controle de parada do temporizador. Este controle é feito através do bit 0 do registrador T1CON. Se estiver em 1 o temporizador está ativo e se estiver em 0 o temporizador para a contagem.

Para testar se o tempo do Timer1 acabou ou não, checa-se o bit TMR1IF do registrador PIR1 está ligado. Se estiver em 1 terminou o tempo.

### 3.1.5 CONVERSOR ANALÓGICO/DIGITAL (A/D)

Segundo Microchip (2003), para configurar a leitura da porta A/D deve-se seguir os seguintes passos, conforme o fluxograma da figura.

**FIGURA 4 - FLUXOGRAMA DA CAPTURA A/D**

Na tabela 13 estão representados os registradores usados nesta operação e seus respectivos bits de controle.

**TABELA 13 - REGISTRADORES PARA OPERAÇÃO DE CAPTURA A/D**

Endereço	Nome	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1FH	ADCON1	ADCS1	ADCS2	CHS2	CHS1	CHS0	GO/DONE		ADON
9FH	ADCON2	ADFM				PCF3	PCFG2	PCFG1	PCFG0
1EH	ADRESH	Bits mais significativos da conversão A/D							
9EH	ADRESL	Bits menos significativos da conversão A/D							
OCH	PIR		ADIF						
8CH	PIE		ADIE						
8BH	INTCON	GIE	PEIE						

### 3.1.6 UTILIZANDO A MEMÓRIA

O PIC possui quatro bancos de memória endereçáveis configurados através dos pinos RP1 e RP0 do registrador STATUS representado na tabela 14. Para mudar de banco de memória precisa-se configurar os pinos RP1 e RP0 do registrador de STATUS conforme demonstrado na tabela 15.

**TABELA 14 - REGISTRADOR DE STATUS**

Endereço	Nome	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
03H	STATUS		RP1	RP0					

**TABELA 15 - CONFIGURAÇÃO DOS BANCOS DE MEMÓRIA**

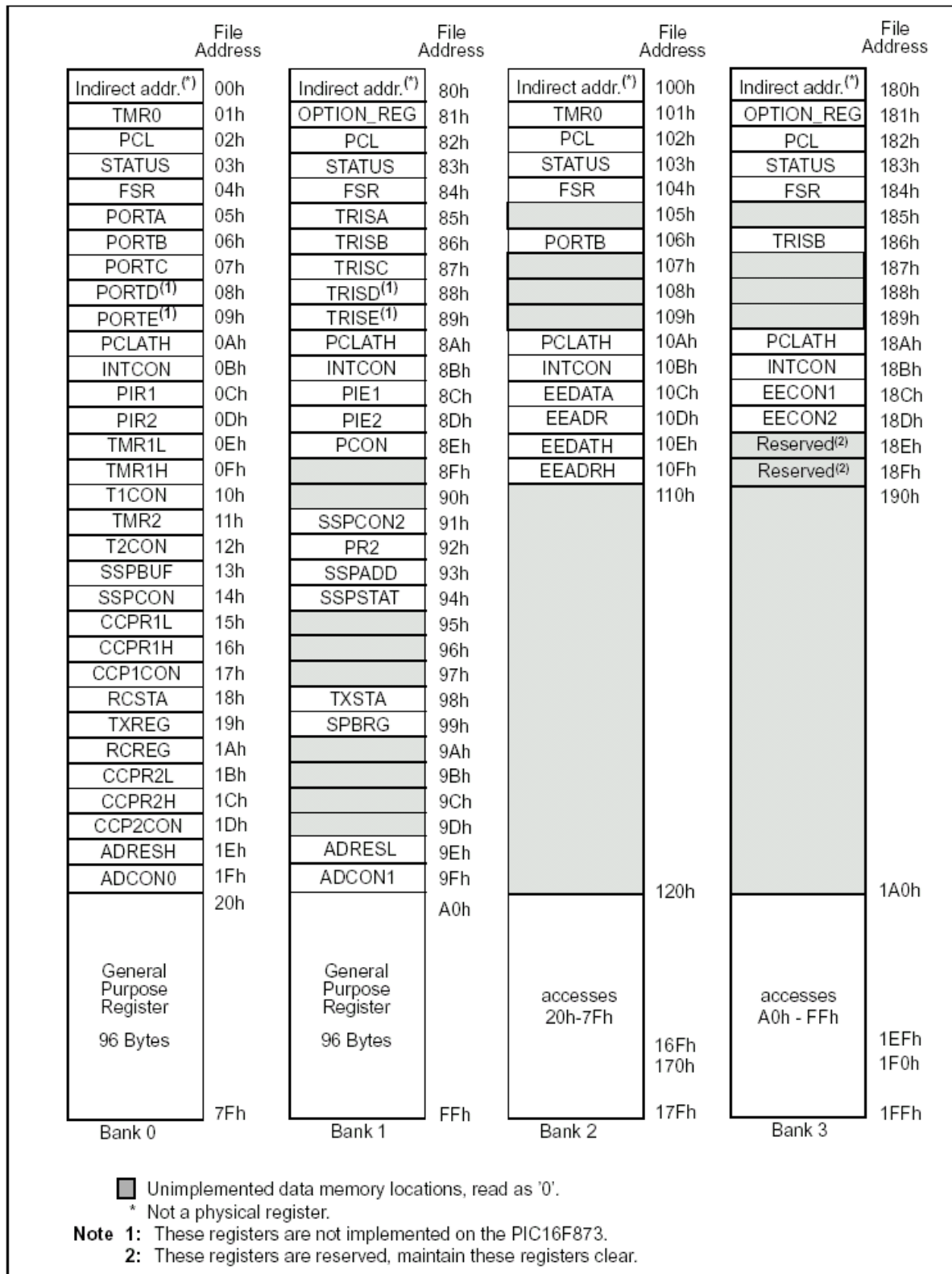
Bits		
RP1	RP0	Banco de memória
0	0	0
0	1	1
1	0	2
1	1	3

No banco 1 de memória a partir do endereço 20H é possível obter 96 bytes de RAM e no banco 0 de memória a partir do endereço A0H mais 96 bytes.

A figura 5, extraída de Microchip (2003), demonstra como está organizada a memória do PIC16F873 e quais memórias estão reservadas para registradores e quais estão disponíveis para o uso.



FIGURA 5 - MAPA DA MEMÓRIA DO PIC16F873

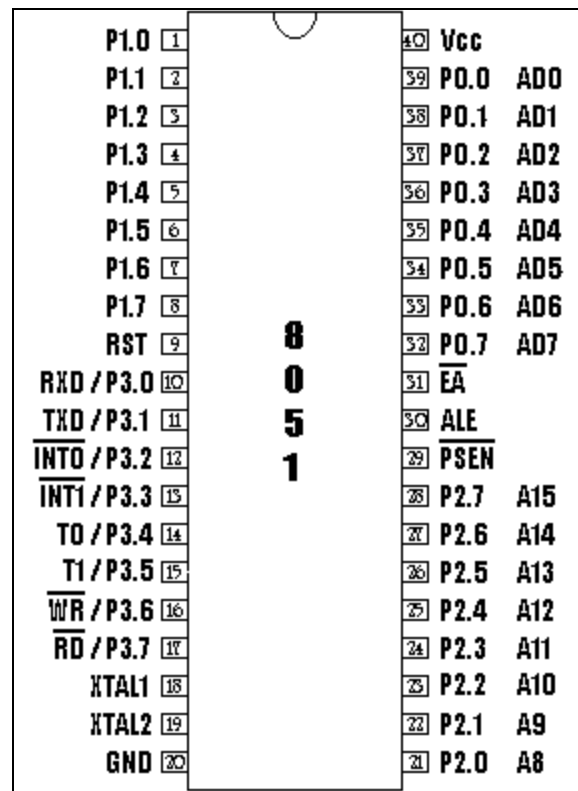


## 3.2 MICROCONTROLADOR 8051

Segundo Lima (2003), o microcontrolador 8051, da Intel, é, sem dúvida, o microcontrolador mais popular atualmente. O dispositivo em si é um microcontrolador de 8 bits relativamente simples, mas com ampla aplicação. O 8051 utiliza tipicamente um clock de 12 MHz, com tempos de execução de cada instrução variando entre 1 $\mu$ s e 4 $\mu$ s.

A figura 6, apresenta a pinagem do 8051

FIGURA 6 - PINAGEM DO 8051.



Além do microprocessador, um sistema básico como este tem os seguintes elementos:

- Interrupções*: são entradas a partir de um sinal externo que fazem com que o processamento seja interrompido e seja iniciada uma subrotina específica.

- b) *Gerador de Reset*: responsável por inicializar o sistema ao ligar ou quando acionado.
- c) *Gerador de Clock*: gera os pulsos necessários ao sincronismo do sistema.
- d) *Memória de Programa*: memória onde o microprocessador vai procurar as instruções a executar. Em sistemas dedicados costuma-se utilizar memórias ROM, embora em alguns casos memórias RAM também sejam utilizadas.
- e) *Memória de Dados*: memória onde o microprocessador lê e escreve dados durante a operação normal. Geralmente é do tipo volátil, embora memórias não-voláteis possam ser utilizadas.
- f) *Seleção de Endereços*: lógica para escolher qual memória ou periférico o microprocessador vai utilizar.
- g) *Portas de I/O*: sua função é a comunicação com o mundo externo. Através delas dispositivos como teclados, impressoras, displays, entre outros, comunicam-se com o sistema.

Este microcontrolador possui um complexo número de instruções que estão listadas no anexo 2.

### **3.2.1 SFR'S (REGISTRADORES ESPECIAIS)**

Segundo Lima (2003), os Registradores de Função Especial (SFRs - *Special Function Registers*) são responsáveis pela maior partes do controle do 8051. Os mesmos estão listados na tabela 11, sendo que alguns deles possuem bits endereçáveis. Alguns dos bits endereçáveis possuem inclusive um nome mnemônico, para maior facilidade de desenvolvimento de software em compiladores.

**TABELA 16 - REGISTRADORES DE FUNÇÃO ESPECIAL DO 8051**

80h	P0	Registrador de estado da P0
90h	P1	Registrador de estado da P1
A0	P2	Registrador de estado da P2
BO	P3	Registrador de estado da P3
88H	TCON	Registro dos Temporizadores/Contadores. Permitem a programação dos mesmos.
89H	TMOD	Registro dos Temporizadores/Contadores. Permitem a programação dos mesmos.
8AH	TL0	Registro de configuração dos bits menos significativos do T0
8CH	TH0	Registro de configuração dos bits mais significativos do T0
8BH	TL1	Registro de configuração dos bits menos significativos do T1
8DH	TH1	Registro de configuração dos bits mais significativos do T1
A8H	IE	Registro para programação das interrupções
B8H	IP	Registro para programação das interrupções.
D0H	PSW	( <i>Program Status Word</i> - palavra de status do programa) é o registro dos <i>Flags</i> do 8051
E0H	ACC	É o acumulador.
F0H	B	Registro auxiliar B.

A seguir, serão mostrados com mais detalhes de alguns dos registradores de funções especiais.

### 3.2.2 PORTAS DE ENTRADA E SAÍDA

Diferentemente do microcontrolador PIC16F873, as portas do 8051 já são de três estados, ou seja, não há necessidade de configurar nenhum registrador para determinar a direção da porta. Basta executar qualquer leitura ou escrita nos registradores P0, P1, P2, P3.

### 3.2.3 TEMPORIZADORES

Para utilizar os temporizadores é necessário configurar os registradores TCON (registrador de controle) e TMOD (registrador de modo de operação). Conforme a configuração eles se incrementam automaticamente até ocorrer o final do tempo selecionado que é indicado pelo TF1.

A tabela 17 demonstra os bits de configuração os registradores TCON e TMOD e a tabela 18 demonstra como configurá-los.

TABELA 17 - REGISTRADORES USADOS NA CONFIGURAÇÃO DOS TEMPORIZADORES

Endereço	Nome	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
89H	TMOD	GATE	C/T	M1	M0	GATE	C/T	M1	M0
88H	TCON	TF1	TR1	TF0	TR0				

TABELA 18 - CONFIGURAÇÃO DOS TEMPORIZADORES

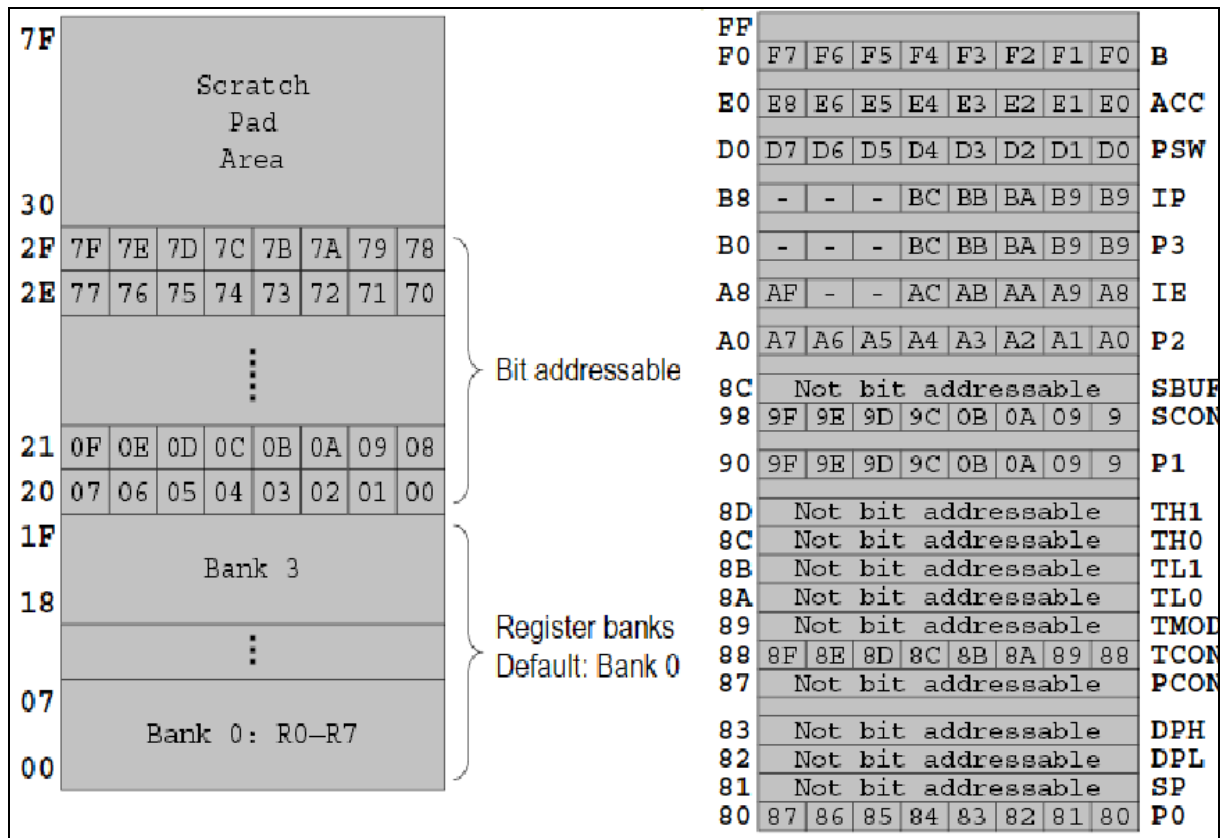
Bits		Função
Gate		aciona o disparo por interrupção
C/T		modo contador ou temporizador
M1	M0	Modo de operação
0	0	Contador de 13 bits
0	1	Contador de 16 bits
1	0	Contador de 8 bits com auto-recarga
1	1	Timer misto
TF1		Flag de estouro temporizador 1
TR1		Controle do temporizador 1
TF0		Flag de estouro do temporizador 0
TR0		Controle do temporizador 0

### 3.2.4 UTILIZANDO MEMÓRIA

O microcontrolador 8051 possui uma RAM interna de 256 bytes sendo a memória de dados vai de 7FH a 00H ou 128 bytes. O resto da memória está alocado para os registradores de funções especiais.

As posições de memória de 20H até 2F são endereçáveis por bit. A figura 7 apresenta o mapa de memória do 8051.

FIGURA 7 -MAPA DE MEMÓRIA DO 8051



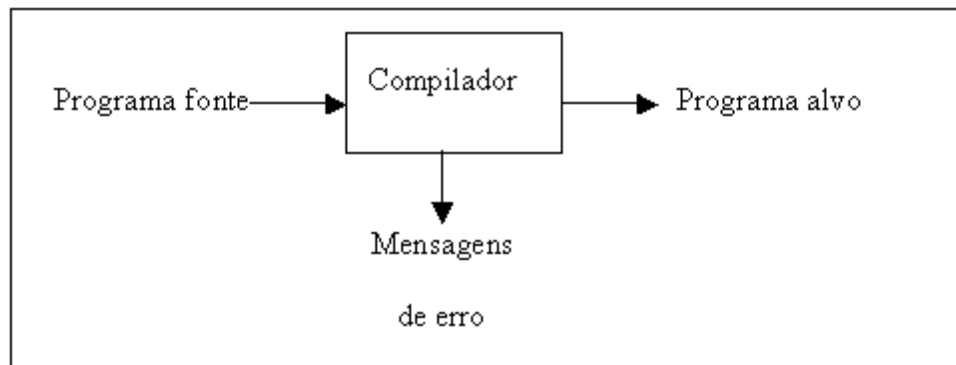
## 4 COMPILADORES

Segundo Aho (1995), um compilador é um programa que lê um programa escrito numa linguagem – a linguagem fonte – e o traduz num programa equivalente em outra linguagem – a linguagem alvo.

Os compiladores são algumas vezes classificados como de uma passagem, de passagens múltiplas, de carregar e executar, depuradores ou otimizantes.

A figura 8 representa um diagrama de um compilador.

FIGURA 8 - DIAGRAMA DE UM COMPILADOR



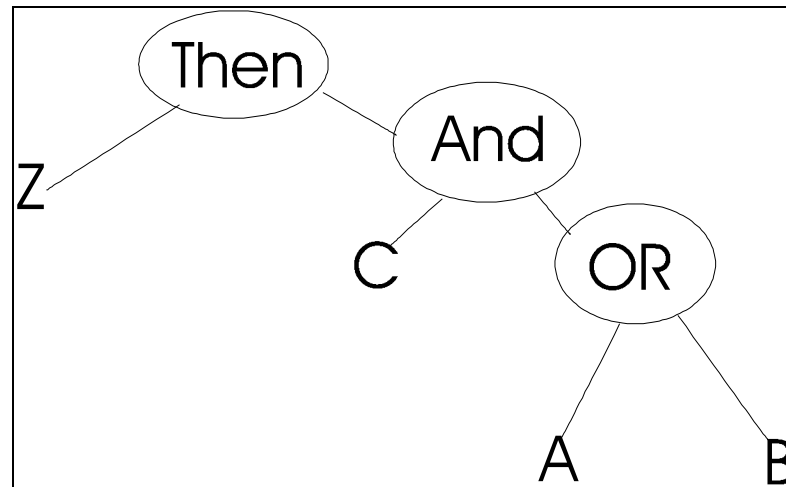
### 4.1 MODELO DE COMPILAÇÃO DE ANÁLISE E SÍNTESE

Segundo Aho (1995), existem duas partes na compilação: a análise e a síntese. A parte de análise divide o programa em partes constituintes e cria uma representação intermediária do mesmo. A de síntese constrói o programa alvo desejado a partir da representação intermediária.

Para exemplificar vamos exemplificar a compilação da seguinte operação lógica IF (A or B) and C then Z. Como a maioria das linguagens assembly trabalha com 2 operandos precisa-se dividir esta instrução de 4 operandos em instruções de 2 operandos.

Como pode ser observar na figura 9 a árvore foi dividida em instruções e em operandos, sendo os operandos, folhas da árvore e as instruções, nós da árvore.

FIGURA 9 - ÁRVORE DA OPERAÇÃO IF (A OR B) AND C THEN Z



Já o processo de síntese consiste em transformar esta árvore em programa alvo para a linguagem correspondente. O quadro 1 ilustra a tradução da árvore da figura 8 para uma linguagem alvo de três operandos. Cada nó da árvore torna-se uma variável temporária e a árvore é analisada das folhas até chegar a raiz que representa o resultado da instrução conforme demonstrado no quadro 1.

QUADRO 1 – TRADUÇÃO DA EXPRESSÃO IF (A OR B) AND C THEN Z

```

Temp1:=A or B

Temp2:=Temp1 and C

Z:=temp2
  
```

## 4.2 BNF OU GRAMÁTICA LIVRE DE CONTEXTO

Segundo Aho (1995), uma gramática de livre contexto ou BNF (forma de Backus-Naur) é utilizada para representar a sintaxe de uma linguagem. Além de especificar a sintaxe da linguagem, pode ser usada como auxílio para guiar a tradução do programa.

Uma gramática de livre contexto possui quatro componentes.



- a. Um conjunto de *tokens* (palavras-chaves como *while*, por exemplo), conhecidos como símbolos terminais;
- b. Um conjunto de não-terminais;
- c. Um conjunto de produções, onde uma produção consiste de um não-terminal, chamado de lado esquerdo da produção, uma seta e uma seqüência de *tokens* e/ou não-terminais, chamados de lado direito da produção.
- d. Uma designação a um dos não-terminais como símbolo de partida.

Para melhor entendimento convencionou-se que os símbolos não-terminais serão itálicos, os *tokens* formados por cadeias de caracteres em negrito e os outros tokens, como dígitos e operandos aritméticos e lógicos, sem itálico ou negrito.

O quadro 2 representa uma gramática para lista de dígitos separados por sinais de mais ou de menos.

**QUADRO 2 – GRAMÁTICA DE OPERAÇÕES MATEMÁTICAS**

*Lista* -> *lista* | *sinal* | *dígito* | &

*Sinal* -> +|-

*Dígito* -> 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

### 4.3 TRADUÇÃO DIRIGIDA PELA SINTAXE

Segundo Aho (1995), pode se obter a tradução dirigida pela sintaxe, associando informações a uma construção de linguagem de programação atrelando os atributos aos símbolos gramaticais que representam a construção. Os valores para os atributos são computados através de regras semânticas associadas às produções da gramática.

A avaliação das regras semânticas pode gerar código, salvar informações numa tabela de símbolos, emitir mensagens de erro ou realizar quaisquer outras atividades.

O t3pico seguinte apresentar3 um exemplo mais detalhado.

## 4.4 GRAM3TICA PARA EXPRESS3ES L3GICAS

Nas linguagens programa33o, inclusive na *Ladder*, as express3es l3gicas s3o usadas para computar valores l3gicos e tamb3m para fluxo de controle.

Segundo Aho (1995), as express3es l3gicas s3o compostas por operadores l3gicos (*and* (e), *or* (ou) e *not* (n3o)) aplicados a elementos que s3o vari3veis l3gicas ou express3es condicionais.

As express3es l3gicas deste cap3tulo seguem a seguinte gram3tica.

$E \rightarrow E \text{ or } E \mid E \text{ and } E \mid \text{not } E \mid \text{true} \mid \text{false}$

Pode-se traduzir uma express3o l3gica sem gerar qualquer c3digo para quaisquer dos operadores l3gicos e sem ter c3digo para avaliar a express3o por inteiro. 3 poss3vel se avaliar as express3es l3gicas sem gerar c3digo para os operadores l3gicos se representar-se o valor de uma express3o por uma posi33o na seq3ncia do c3digo.

Abaixo um esquema de tradu33o para express3es l3gicas para uma linguagem com tr3s operandos. Onde a esquerda est3 a regra de produ33o e a direita est3 regra sem3ntica. Quando a express3o se encaixar na regra de produ33o o programa ativar3 a regra sem3ntica correspondente.

TABELA 19 - ESQUEMA DE TRADU33O PARA UM LINGUAGEM DE TR3S OPERANDOS

Produ33o	Regra sem3ntica
$E \rightarrow E_1 \text{ or } E_2$	{E.local:= novotemporario; Emitir (E.local ':=' E <sub>1</sub> .local 'or' E <sub>2</sub> .local)}
$E \rightarrow E_1 \text{ and } E_2$	{E.local:= novotemporario; Emitir (E.local ':=' E <sub>1</sub> .local 'and' E <sub>2</sub> .local)}
$E \rightarrow \text{not } E_1$	{E.local:= novotemporario; Emitir (E.local ':=' not' E <sub>1</sub> .local)}
$E \rightarrow (E_1)$	{E.local:= E <sub>1</sub> .local}
$E \rightarrow \text{true}$	{E.local:= novo_temporario Emitir (E.local ':=' 1')}
$E \rightarrow \text{false}$	{E.local:= novo_temporario Emitir (E.local ':=' 0')}

A expressão (a **or** b) **and** (c **or** d), geraria a tradução apresentada no quadro 3. Ao encontrar a primeira operação OR o programa ativaria a regra semântica correspondente. Geraria um novo temporário indicado por T0 no código gerado abaixo na linha 1. Substituiria esta variável temporária no código continuando a tradução. Ao identificar a segunda operação OR geraria o temporário T1, e emitiria a linha 2. Substituiria a variável no código e continuaria. Ao encontrar a operação AND terminaria a tradução guardando o resultado na variável T2.

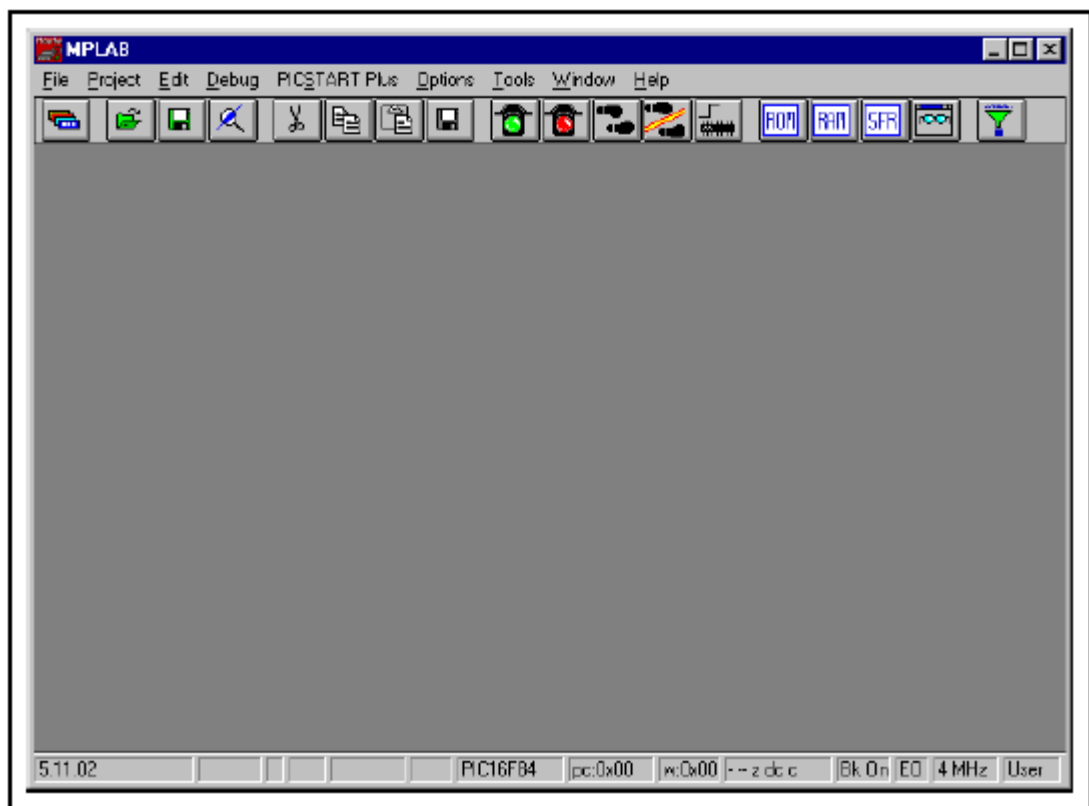
**QUADRO 3 - TRADUÇÃO DA EXPRESSÃO (A OR B) AND (C OR D)**

1 : T0 := a or b	(T0) <b>and</b> (c or d)
2: T1:= c or d	(T0) <b>and</b> (T1)
3: T2:=T0 and T1	

## 5 MPLAB

O MPLAB é uma ferramenta desenvolvida pela Microchip© para editar, simular e montar os códigos *assembly* para os microcontroladores fabricados pela empresa. A ferramenta possibilita editar o código através da edição de um arquivo texto. Este arquivo texto pode também ser compilado e pode ser testado passo-a-passo. A figura 10 exibe a interface do MPLAB.

FIGURA 10 - INTERFACE DO MPLAB



Para poder compilar e testar um programa assembly no MPLAB é necessário primeiramente criar um projeto. Para criar novo projeto clique em *Project/New Project* e salve com o nome que desejar.

Depois de escolhido o nome do projeto, o programa exibe a tela de edição de projeto conforme ilustrado na figura. No campo *Target Filename*, coloque o nome do arquivo hexadecimal que será gerado. No campo *Development Mode* indique com qual

microcontrolador da Microchip está trabalhando. Clique em *Add Node* e selecione o arquivo que se deseja compilar e simular.

FIGURA 11 - TELA DE EDIÇÃO DO PROJETO



Depois de executados os passos anteriores, para compilar o programa clique em *Debug/Run/Execute*. Para interromper a compilação clique em *Debug/Run/Reset*.

Para testar passo-a-passo clique em *Debug/Run/Step*. No modo passo-a-passo o programa executará cada linha do programa. Para acompanhar o estado da memória de programa clique em *Window/Program Memory*. Para acompanhar o estado dos registradores especiais clique em *Window/Special Function Registers*. Para acompanhar o estado dos registradores clique em *Window/File Registers*.

## 6 DESENVOLVIMENTO DO TRABALHO

Depois de levantados as informações sobre os microcontroladores, linguagem Ladder e compiladores partiu-se para especificação do protótipo, utilizando a ferramenta Rational Rose e depois se programou em Object Pascal na ferramenta RAD, Borland Delphi 5.0.

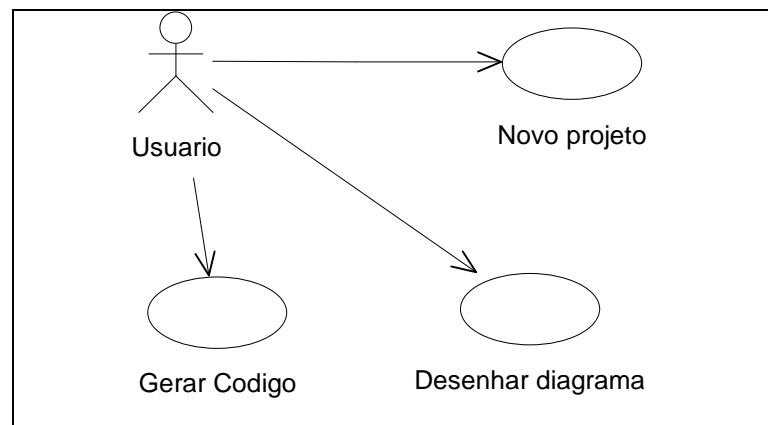
O sistema deve ser capaz de desenhar um diagrama Ladder e seus símbolos, obter os dados do microcontrolador selecionado pelo sistema, e também a sua sintaxe. Com esses dados o programa deve gerar um código *assembly* correspondente ao microcontrolador escolhido.

A especificação foi feita em UML (*unified modeling language*) utilizando a ferramenta Rational Rose da Rational©.

### 6.1 ESPECIFICAÇÃO

O diagrama da figura 12, representa os casos de usos mais relevantes para o usuário do sistema, que são criar novo projeto, desenhar o diagrama *Ladder*, e gerar o código a partir do diagrama *Ladder*.

FIGURA 12 - DIAGRAMA DE CASOS DE USO



As figuras 13, 14 e 15 representam o diagrama de classes do projeto.

FIGURA 13 - DIAGRAMA DE CLASSES

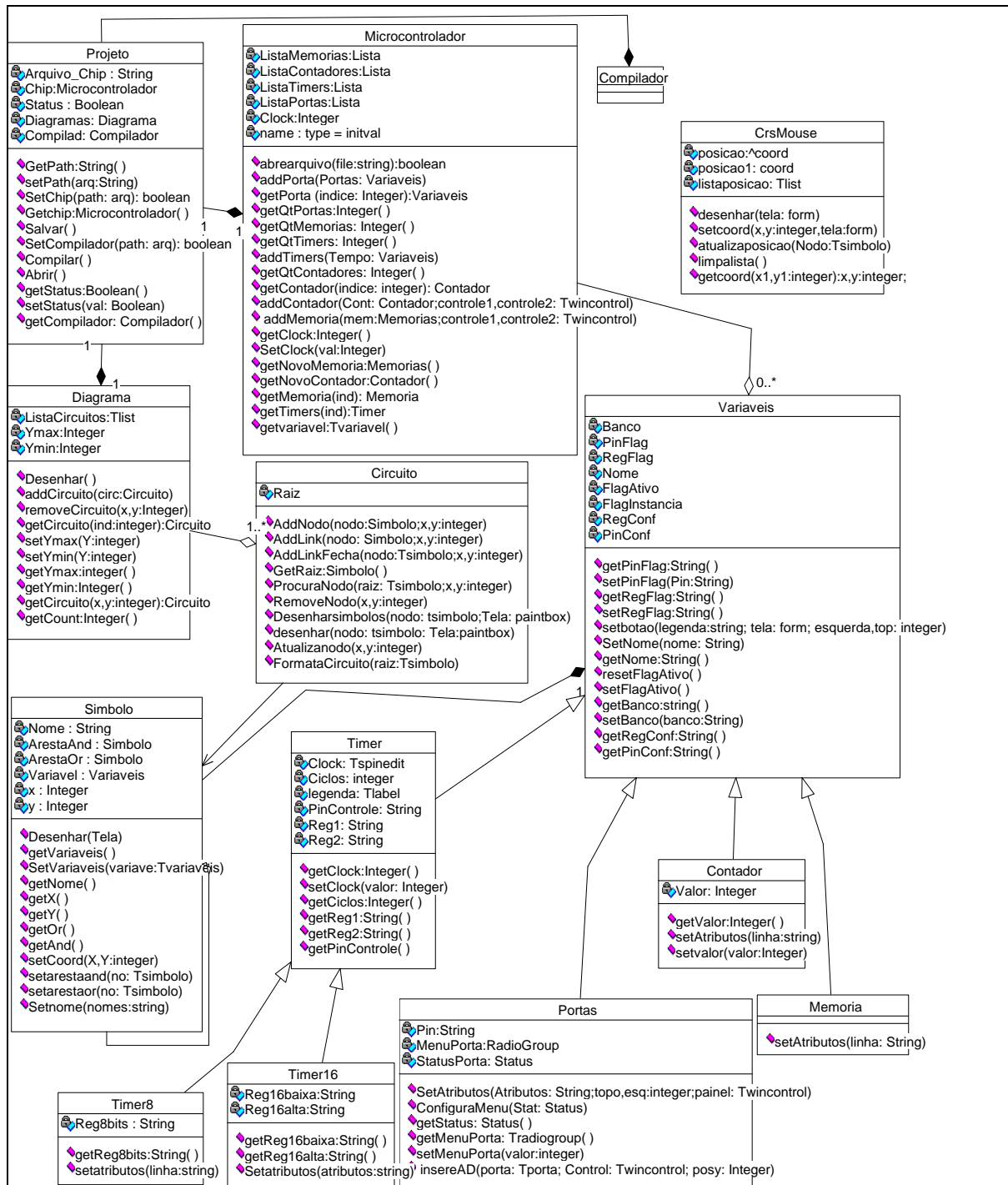


FIGURA 14 - DIAGRAMA DA CLASSE SÍMBOLO

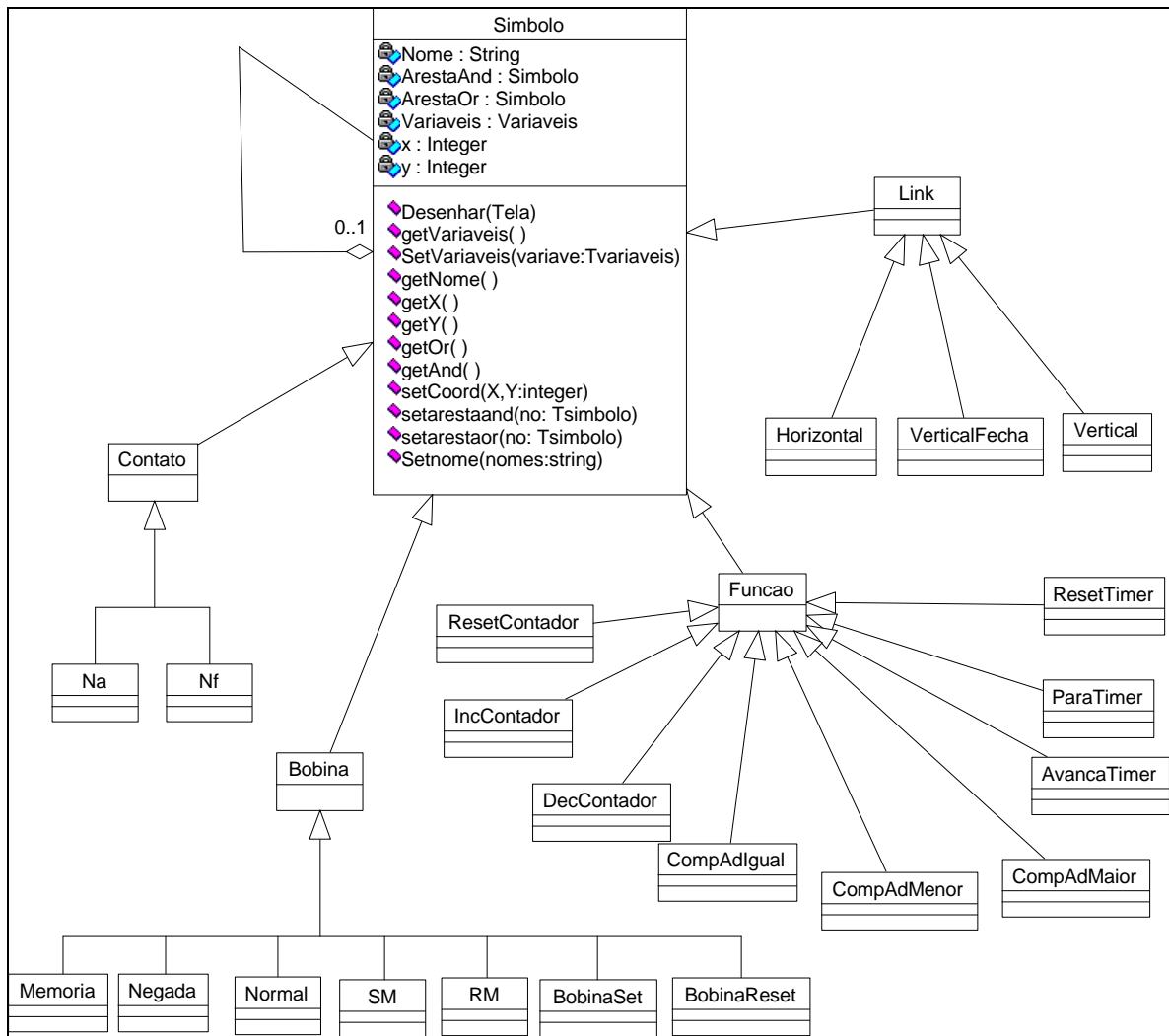
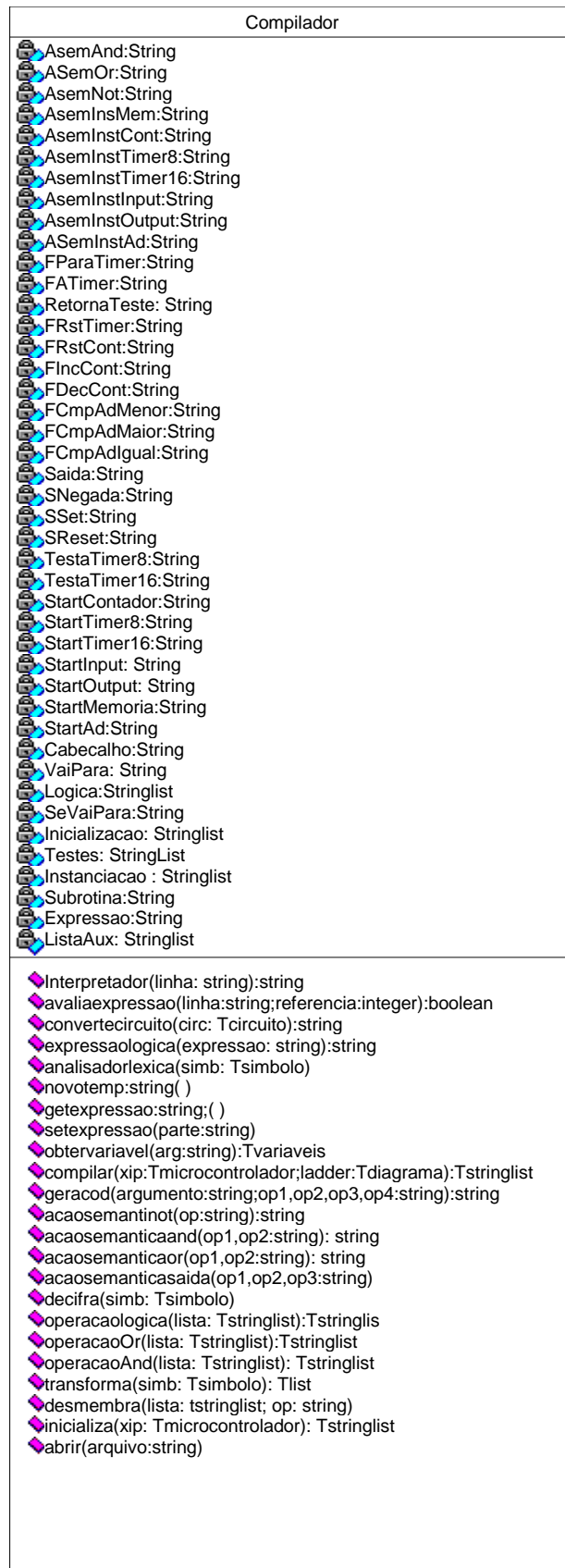




FIGURA 15 - DIAGRAMA DA CLASSE COMPILADOR



Estas são as principais classes do projeto.

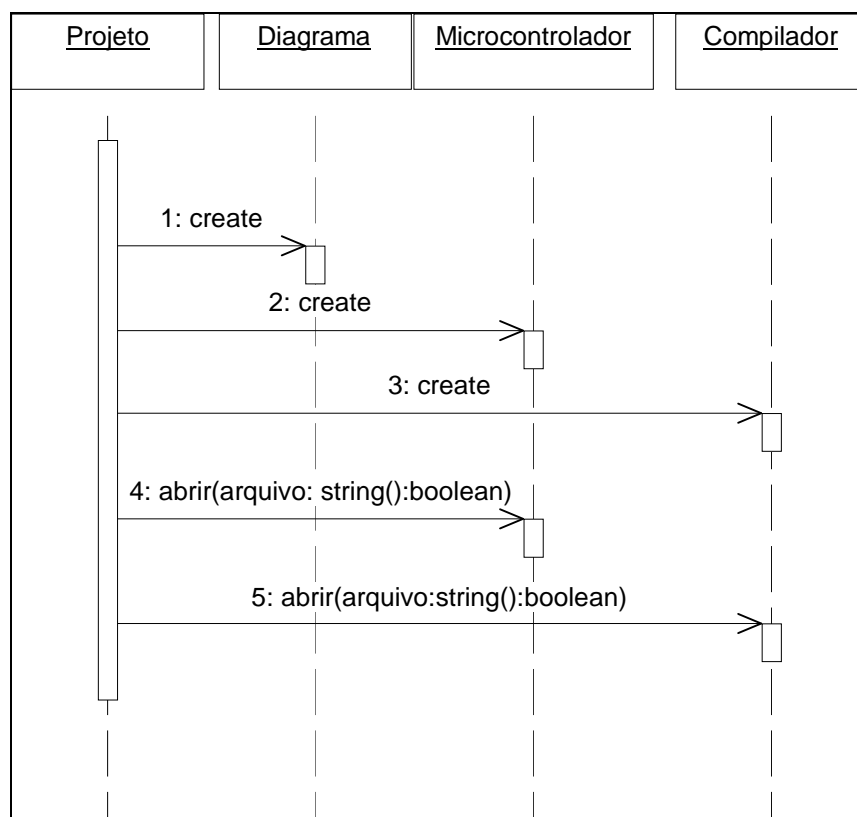
- a) Classe TProjeto – É a classe principal da aplicação. Responsável por gerenciar as demais classes e por gerenciar o projeto. É composta por um diagrama, um compilador e um microcontrolador como pode ser visto na figura 12.
- b) Classe TMicrocontrolador - esta classe tem a função de configurar e gerenciar as propriedades dos microcontroladores e é composto de variáveis que podem ser portas, timers, contadores ou memórias.
- c) Classe TVariaveis - classe abstrata que representa as variáveis dos microcontroladores. Esta classe serve de molde para os demais elementos componentes de um microcontrolador. Pode se especializar em Tporta, Tmemoria, Ttimer ou Tmemória.
- d) Classe TPorta - classe que representa uma porta de entrada e saída de um microcontrolador. Pode ser entrada, saída ou entrada analógica/digital.
- e) Classe TTimer - classe que representa um temporizador de um microcontrolador.
- f) Classe TContador - classe que representa propriedades e métodos de um contador de 8 bits.
- g) Classe TMemoria - classe que representa propriedades e métodos de uma memória de 8 bits .
- h) Classe TDiagrama - representa um contêiner de circuitos. Esta classe tem como função gerenciar e configurar as propriedades pertinentes ao diagrama Ladder e adicionar e remover circuitos.
- i) Classe TCircuito - representa um subconjunto de símbolos do diagrama Ladder. É responsável por gerenciar os símbolos e a forma como se apresentam na tela.
- j) Classe TSimbolo - classe abstrata que serve de molde para os demais símbolos do diagrama Ladder. Pode se especializar em Tcontato, Tbobinas, Tfuncao e Tlink.
- k) Classe TContato - classe que representa contatos da linguagem ladder. Pode-se especializar em TcontatoNA, TcontatoNF, TcontatoSTP, TcontatoSTN
- l) Classe TBobinas - classe que representa bobinas da linguagem ladder.
- m) Classe TFuncao - classe que representa funções da linguagem ladder, pode se especializar em .
- n) Classe TLink - classe que representa elementos de conexão da linguagem Ladder. Pode se especializar em TlinkVerticalFecha, TlinkVerticalAbre e TlinkHorizontal.

- o) Classe TCompilador – classe responsável por operações de um compilador, como análise léxica, tradução de expressões.

A figura 16 representa a seqüência de operações executadas quando se inicia um novo projeto. Primeiro cria-se a classe diagrama, a classe microcontrolador e a classe compilador. Depois se abre o arquivo de configuração do microcontrolador e obtêm-se os parâmetros de configuração e linguagem do mesmo.

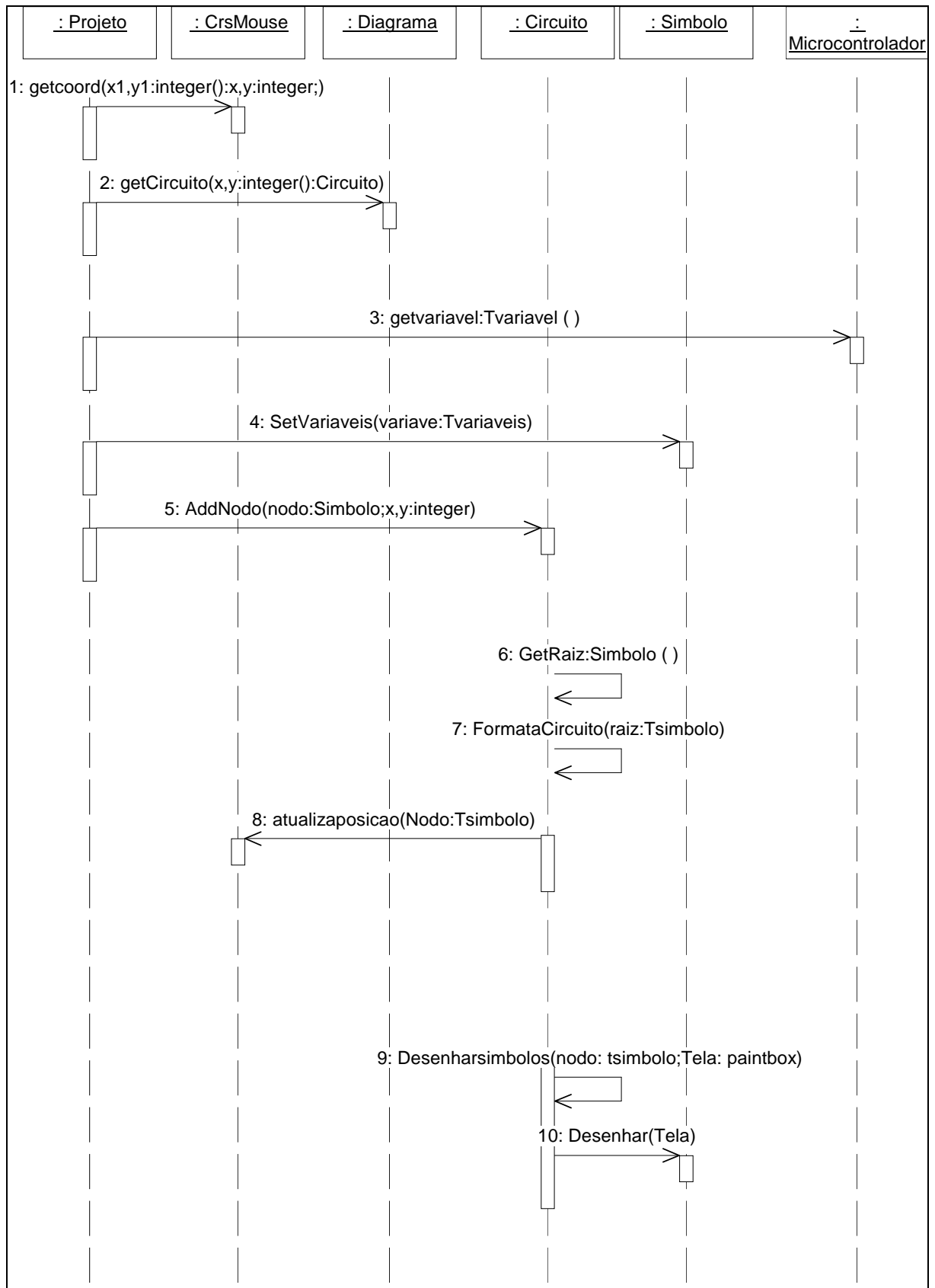
Os arquivos de configuração dos microcontroladores estão listados nos anexos 3 e 4.

**FIGURA 16 - DIAGRAMA DE SEQÜÊNCIA DE ESCOLHER MICROCONTROLADOR**



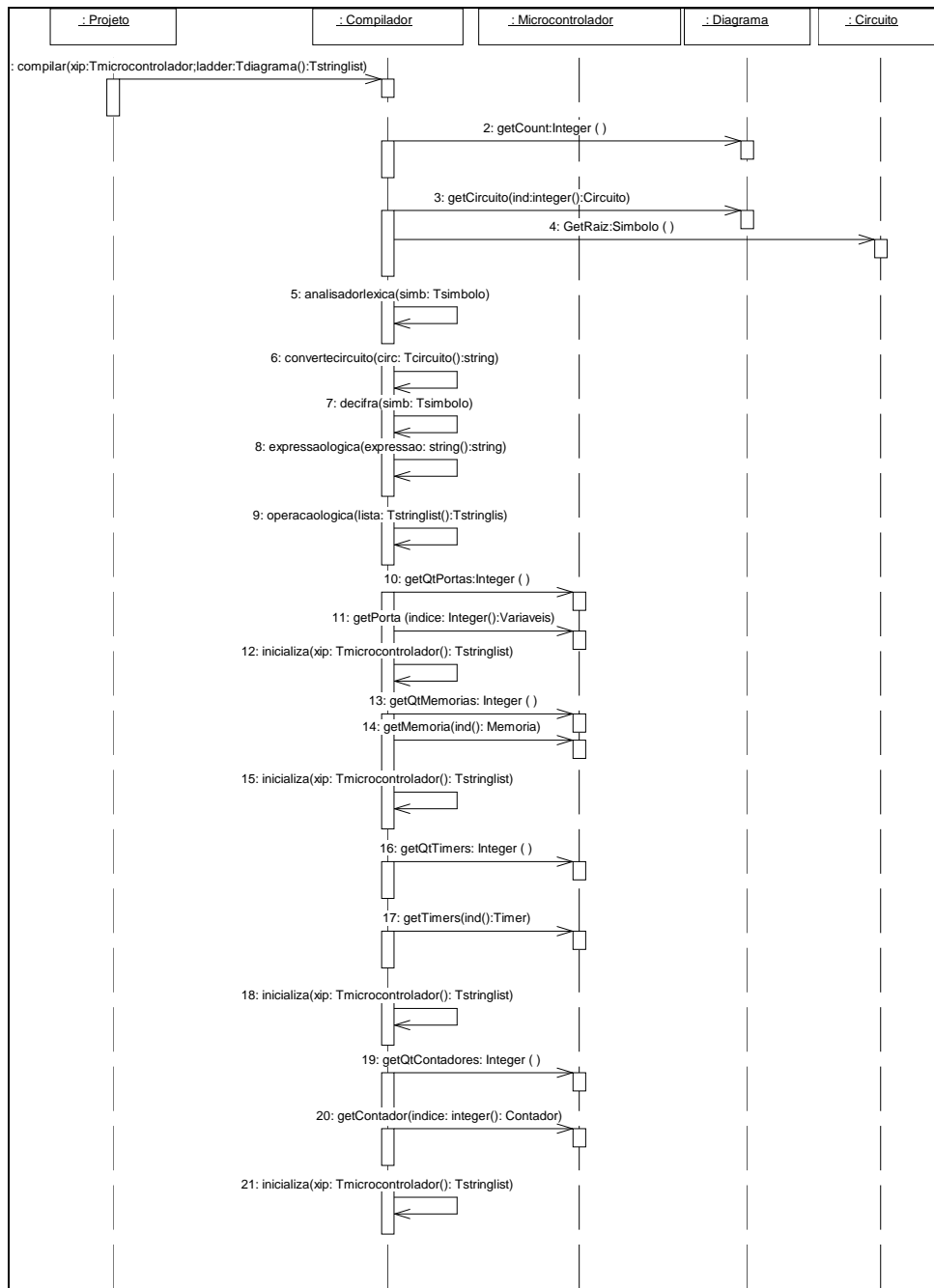
A figura 17 representa a seqüência de operações executadas quando se insere um símbolo no diagrama *Ladder*. Obtém-se a coordenada do cursor. Descobre-se o circuito onde está o cursor. Seleciona-se uma variável do microcontrolador. Então adiciona o símbolo ao circuito e atualiza o diagrama depois do elemento inserido.

**FIGURA 17 - DIAGRAMA DE SEQUÊNCIA DE DESENHAR DIAGRAMA**



A figura 18 demonstra como é feita a geração de código. Primeiramente a classe Projeto solicita a classe compilador a compilação do código. A classe compilador verifica se o diagrama está bem construído através da análise léxica. Se estiver correto converte-se o circuito em operações lógicas e depois se traduz para a linguagem assembly correspondente até chegar ao último circuito.

**FIGURA 18 - DIAGRAMA DE SEQÜÊNCIA DE GERAR CÓDIGO**



## 6.2 ESQUEMAS DE TRADUÇÕES

A linguagem Ladder baseia-se na avaliação dos circuitos através de operações lógicas. Para traduzir expressões lógicas de um linguagem alvo para uma linguagem objeto torna-se mais fácil usando esquemas de traduções.

A partir de uma regra de produções encontrada na linguagem alvo aplicam-se as regras semânticas correspondentes.

Este projeto utiliza os seguintes esquemas de tradução de expressões de lógicas conforme apresentado na tabela 20 para o PIC16F873 e na tabela 21 para o 8051.

TABELA 20 - ESQUEMA DE TRADUÇÃO PARA O PIC16F873

<b>Produção</b>	<b>Regra semântica</b>
$E \rightarrow E_1 \text{ or } E_2$	<i>{E.local:= novotemporario; Emitir ('BCF E.Local BTFSC E1.Local BSF E.Local BTFSC E2.Local BSF E.Local)}</i>
$E \rightarrow E_1 \text{ and } E_2$	<i>{E.local:= novotemporario; Emitir (BSF E.local BTFSS E1.Local BCF E.Local BTFSS E2.Local BCF E.local)}</i>
$E \rightarrow \text{not } E_1$	<i>{E.local:= novotemporario; Emitir ('BTFSC E1.local BCF E.Local BTFSS E1.Local BSF E.Local)}</i>
$E \rightarrow (E_1)$	<i>{E.local:= E1.local}</i>

TABELA 21 - ESQUEMA DE TRADUÇÃO PARA O 8051

<b>Produção</b>	<b>Regra semântica</b>
$E \rightarrow E_1 \text{ or } E_2$	{ <i>E.local</i> := novotemporario; Emitir ('MOV C,'E1.Local 'ORL C,'E2.Local 'MOV 'E.local',C')}
$E \rightarrow E_1 \text{ and } E_2$	{ <i>E.local</i> := novotemporario; Emitir (MOV C,'E1.Local 'ANL C,'E2.Local 'MOV 'E.local',C')}
$E \rightarrow \text{not } E_1$	{ <i>E.local</i> := novotemporario; Emitir ('MOV C,' E1.local 'CPL C' 'MOV 'E.Local',C')}
$E \rightarrow (E_1)$	{ <i>E.local</i> := <i>E1.local</i> }

## 6.3 OPERACIONALIDADE DA IMPLEMENTAÇÃO

Para demonstrar o funcionamento do protótipo implementou-se um meio-somador, que consiste de uma porta ou-exclusivo, cujas entradas resultam no valor da soma. E uma porta AND, cujas entradas resultam no valor do *Carry*.

### 6.3.1 PRINCIPAIS FUNÇÕES

A figura 16 demonstra a tela do protótipo. As principais funções estão descritas abaixo

0 - Tela onde se insere o circuito e símbolos do diagrama Ladder posicionados através de um cursor que muda de posição com o clique do mouse.

1 - Arquivo – Neste item estão opções para gerenciar o projeto, como abrir, salvar e novo projeto.

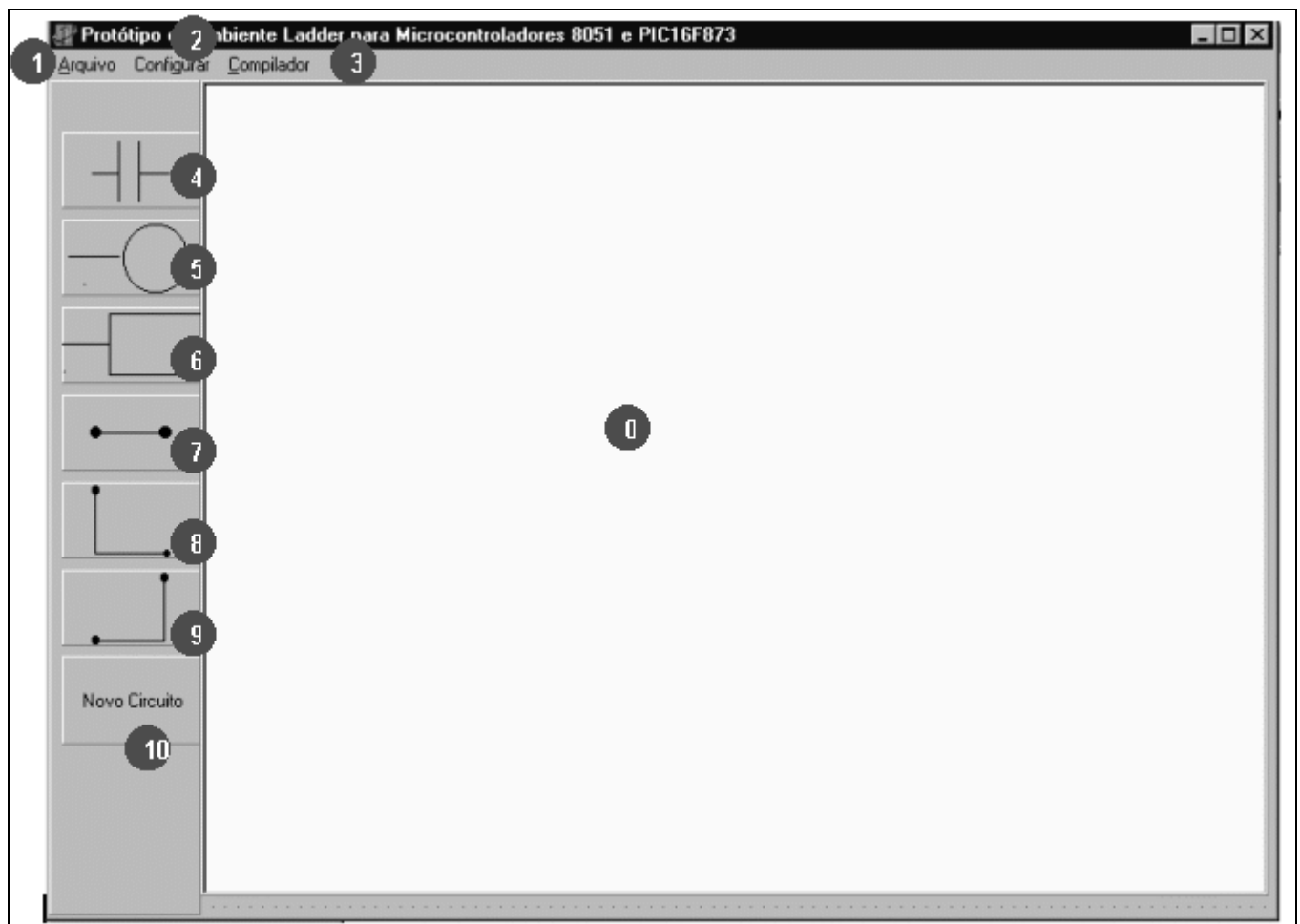
2 - Configurar – Atribuir nomes mnemônicos as variáveis do microcontrolador que serão usadas no diagrama Ladder. No caso de timers e contadores também se configura o valor da temporização e valor limite da contagem. No caso de portas pode se configurar a função da porta.

3 - Compilador – Abre um novo formulário com o código assembly gerado a partir do diagrama Ladder se o diagrama estiver correto.

4 - Contato – Insere um contato na tela, onde estiver o cursor.

- 5 - Bobina – Insere uma bobina na tela, onde estiver o cursor.
- 6 - Função – Insere uma função na tela, onde estiver o cursor.
- 7 - Conexão Horizontal – Inserir uma conexão horizontal na tela, onde estiver o cursor
- 8 - Conexão Vertical – Abre uma conexão vertical no diagrama onde estiver o cursor
- 9 - Conexão Vertical Fecha – Fecha uma conexão vertical no diagrama onde estiver o cursor.

**FIGURA 19 - TELA DE ABERTURA DO PROTÓTIPO**





### 6.3.2 INICIANDO UM NOVO PROJETO

Para iniciar um novo projeto, clique no menu arquivo, e depois clique em novo. Aparecerá um formulário conforme a figura 20 e dentre as opções escolha o microcontrolador desejado, clique em OK e se iniciará um novo projeto. Esta escolha irá determinar o ambiente do programa dinamicamente, atualizando os menus de portas, contadores e memórias. Se não for escolhido um microcontrolador o ambiente ficará desabilitado para edição.

Quando escolhida a opção o programa abrirá o arquivo de configuração que irá atualizar os formulários das variáveis do microcontrolador.

FIGURA 20. TELA DE ESCOLHA DO MICROCONTROLADOR



### 6.3.3 CONFIGURANDO AS VARIÁVEIS

Depois de escolhido o microcontrolador é necessário configurar as variáveis que serão utilizadas no programa, atribuindo uma palavra mnemônica aos elementos do microcontroladores. As variáveis podem ser memórias, contadores, portas de entrada e saída e temporizadores.

Para tal procedimento, clique no menu principal em Configurar. Depois escolha o menu com o tipo correspondente de variável que se quer usar, se a variável é porta, memória, contador ou temporizador. Se escolher portas, aparecerá a tela que está demonstrada na figura 18. À esquerda aparecerá o nome do elemento do microcontrolador.

Para que as variáveis sejam válidas tem que ser colocado um nome na caixa de texto representa na figura 21 por VariavelA e também deve ser selecionada a função da variável selecionando quaisquer dos itens de seleção no menu que aparece ao lado da caixa de texto.

Depois de configurar as variáveis clique OK.

**FIGURA 21 - CONFIGURAÇÃO DE PORTAS**

The image shows a software window titled "Configuração de Portas" (Port Configuration). It contains a list of ports, each with a text input field and a set of radio buttons. The ports are labeled Ra0 through Ra5, Rb0 through Rb7, and Rc0 through Rc1. The "Ra0" port is configured with the text "VariavelA" in its input field and the "Entrada" radio button selected. Other ports have empty input fields and either "Entrada" or "Saida" selected. Ports Rb3 through Rb7 and Rc0 through Rc1 have an additional "ST" radio button option. At the bottom of the window are "OK" and "Cancelar" buttons.

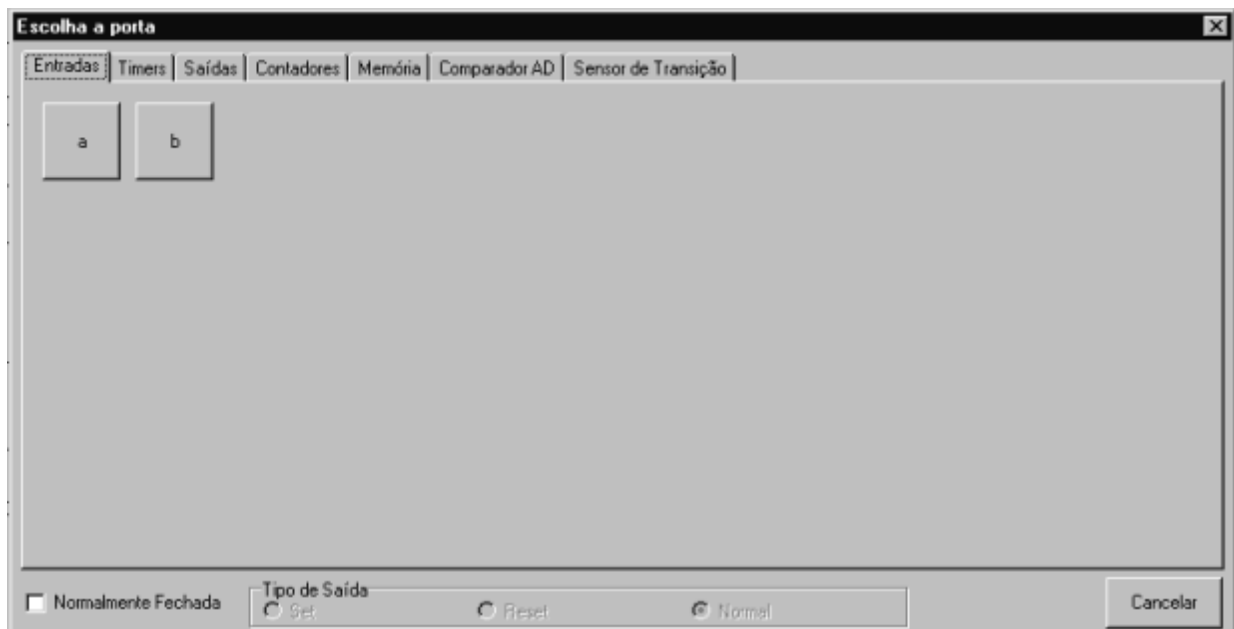
Port	Text Field	Entrada	Saida	A/D	ST
Ra0	VariavelA	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	
Ra1		<input type="radio"/>	<input type="radio"/>		
Ra2		<input type="radio"/>	<input type="radio"/>		
Ra3		<input type="radio"/>	<input type="radio"/>		
Ra4		<input type="radio"/>	<input type="radio"/>		
Ra5		<input type="radio"/>	<input type="radio"/>		
Rb0		<input type="radio"/>	<input type="radio"/>		
Rb1		<input type="radio"/>	<input type="radio"/>		
Rb2		<input type="radio"/>	<input type="radio"/>		
Rb3		<input type="radio"/>	<input type="radio"/>		<input type="radio"/>
Rb4		<input type="radio"/>	<input type="radio"/>		<input type="radio"/>
Rb5		<input type="radio"/>	<input type="radio"/>		<input type="radio"/>
Rb6		<input type="radio"/>	<input type="radio"/>		<input type="radio"/>
Rb7		<input type="radio"/>	<input type="radio"/>		<input type="radio"/>
Rc0		<input type="radio"/>	<input type="radio"/>		
Rc1		<input type="radio"/>	<input type="radio"/>		

### 6.3.4 DESENHANDO O DIAGRAMA

Depois de declaradas as variáveis, pode-se começar o desenho do diagrama. Para desenhar o diagrama use qualquer dos botões à esquerda da tela do protótipo. Clicando no botão dos contatos ou bobinas aparecerá o formulário da figura 22.

Neste formulário, escolha a variável clicando nas orelhas do formulário e depois no botão desejado.

FIGURA 22 - ESCOLHENDO A VARIÁVEL



### 6.3.5 COMPILANDO O DIAGRAMA

Depois do diagrama estar completo conforme a figura 23, o projeto pode ser compilado. Se o circuito não estiver completo o protótipo não permite a compilação.

Para compilar o código clique em Compilador no menu principal. Aparecerá uma tela igual a da figura 24, com o código *assembly* do respectivo microcontrolador. Para salvar o código gerado clique em salvar e escolha um nome.

FIGURA 23 - DIAGRAMA LADDER DE UM MEIO-SOMADOR

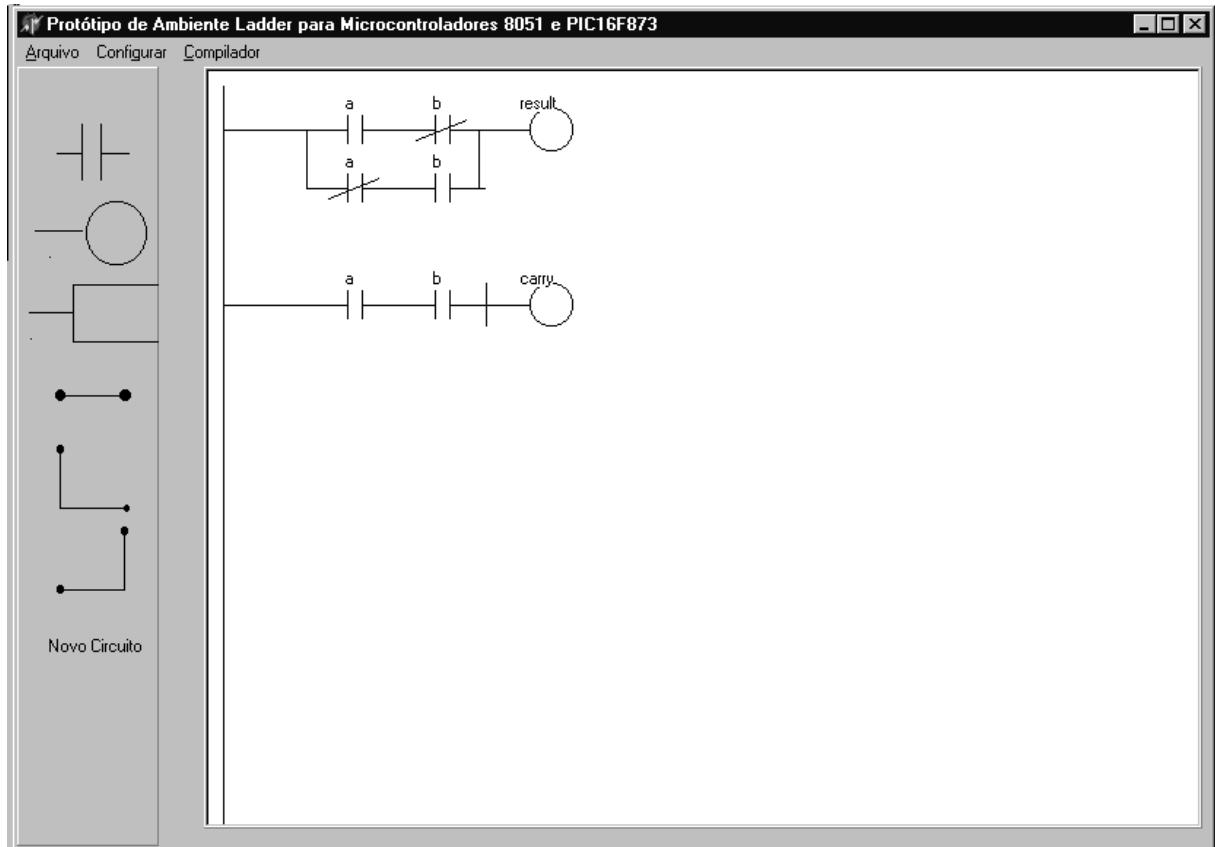
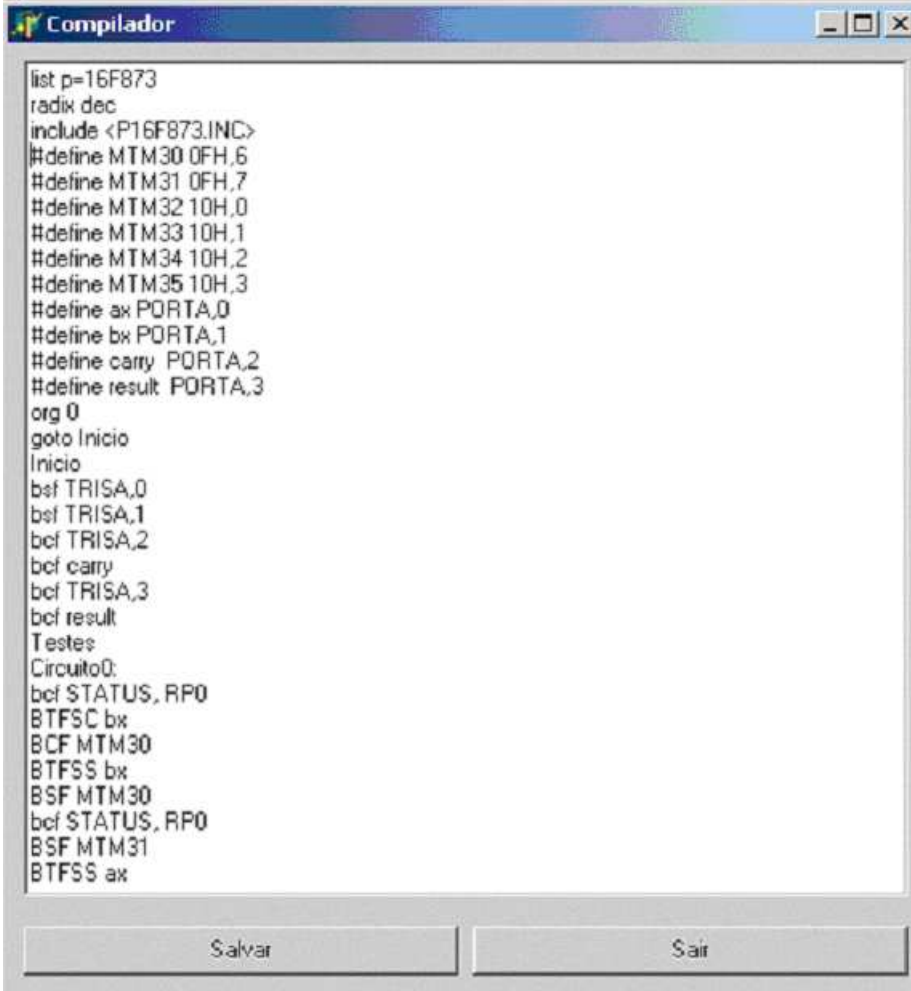


FIGURA 24 -CÓDIGO ASSEMBLY GERADO DO MEIO SOMADOR



```
list p=16F873
radix dec
include <P16F873.INC>
#define MTM30 0FH,6
#define MTM31 0FH,7
#define MTM32 10H,0
#define MTM33 10H,1
#define MTM34 10H,2
#define MTM35 10H,3
#define ax PORTA,0
#define bx PORTA,1
#define carry PORTA,2
#define result PORTA,3
org 0
goto Inicio
Inicio
bcf TRISA,0
bcf TRISA,1
bcf TRISA,2
bcf carry
bcf TRISA,3
bcf result
Testes
Circuito0:
bcf STATUS, RP0
BTFSC bx
BCF MTM30
BTFSS bx
BSF MTM30
bcf STATUS, RP0
BSF MTM31
BTFSS ax
```

### 6.3.6 MONTANDO O CÓDIGO GERADO

Depois do código gerado conforme a figura 24, o projeto pode ser montado. Para montar código do PIC16F873 usou-se o MPLAB.

Abra o MPLAB, crie um novo projeto, conforme explicado no capítulo 5, adicione ao projeto o código gerado e então compile. Aparecerá a tela conforme a figura 25.

Se o código compilar então não apresentou nenhum erro de sintaxe.

FIGURA 25 - RESULTADO DA MONTAGEM

The screenshot displays the MPLAB IDE interface. On the left, the assembly code is visible, listing instructions such as BSF, BCF, and BTFSS for various bit positions (e.g., MTH33, MTH34, MTH35). The main window shows the 'Build Results' pane, which contains a list of messages and warnings. The messages include 'Register in operand not in bank 0' and 'Found opcode in column 1' for various instructions. The build process concludes with the message 'Build completed successfully.' The status bar at the bottom indicates the target device is PIC16F672 and the simulation frequency is 4 MHz.

```

C:\Program Files\MPLAB\MPLAB\EXAMPLES\PJT
Circuito0:
BSF MTH33
BTFSS a
BCF MTH33
BTFSS b
BCF MTH33
BCF MTH34
BTFSC c
BSF MTH34
BTFSC d
BSF MTH34
BSF MTH35
BTFSS MTH33
BCF MTH35
BTFSS MTH34
BCF MTH35
BTFSS MTH35
BCF e
BTFSC MTH35
BSF e
goto Testes
END

Build Results
Message[302] C:\ARQUIVOS\MPLAB\EXAMPLES\ID.ASH 18 : Register in operand not in bank 0. Ensure tha
Warning[203] C:\ARQUIVOS\MPLAB\EXAMPLES\ID.ASH 19 : Found opcode in column 1. (bcf)
Message[302] C:\ARQUIVOS\MPLAB\EXAMPLES\ID.ASH 19 : Register in operand not in bank 0. Ensure tha
Warning[203] C:\ARQUIVOS\MPLAB\EXAMPLES\ID.ASH 20 : Found opcode in column 1. (bcf)
Warning[203] C:\ARQUIVOS\MPLAB\EXAMPLES\ID.ASH 23 : Found opcode in column 1. (BSF)
Warning[203] C:\ARQUIVOS\MPLAB\EXAMPLES\ID.ASH 24 : Found opcode in column 1. (BTFSS)
Warning[203] C:\ARQUIVOS\MPLAB\EXAMPLES\ID.ASH 25 : Found opcode in column 1. (BCF)
Warning[203] C:\ARQUIVOS\MPLAB\EXAMPLES\ID.ASH 26 : Found opcode in column 1. (BTFSS)
Warning[203] C:\ARQUIVOS\MPLAB\EXAMPLES\ID.ASH 27 : Found opcode in column 1. (BCF)
Warning[203] C:\ARQUIVOS\MPLAB\EXAMPLES\ID.ASH 28 : Found opcode in column 1. (BCF)
Warning[203] C:\ARQUIVOS\MPLAB\EXAMPLES\ID.ASH 29 : Found opcode in column 1. (BTFSC)
Warning[203] C:\ARQUIVOS\MPLAB\EXAMPLES\ID.ASH 30 : Found opcode in column 1. (BSF)
Warning[203] C:\ARQUIVOS\MPLAB\EXAMPLES\ID.ASH 31 : Found opcode in column 1. (BTFSC)
Warning[203] C:\ARQUIVOS\MPLAB\EXAMPLES\ID.ASH 32 : Found opcode in column 1. (BSF)
Warning[203] C:\ARQUIVOS\MPLAB\EXAMPLES\ID.ASH 33 : Found opcode in column 1. (BSF)
Warning[203] C:\ARQUIVOS\MPLAB\EXAMPLES\ID.ASH 34 : Found opcode in column 1. (BTFSS)
Warning[203] C:\ARQUIVOS\MPLAB\EXAMPLES\ID.ASH 35 : Found opcode in column 1. (BCF)
Warning[203] C:\ARQUIVOS\MPLAB\EXAMPLES\ID.ASH 36 : Found opcode in column 1. (BTFSS)
Warning[203] C:\ARQUIVOS\MPLAB\EXAMPLES\ID.ASH 37 : Found opcode in column 1. (BCF)
Warning[203] C:\ARQUIVOS\MPLAB\EXAMPLES\ID.ASH 38 : Found opcode in column 1. (BTFSS)
Warning[203] C:\ARQUIVOS\MPLAB\EXAMPLES\ID.ASH 39 : Found opcode in column 1. (BCF)
Warning[203] C:\ARQUIVOS\MPLAB\EXAMPLES\ID.ASH 40 : Found opcode in column 1. (BTFSC)
Warning[203] C:\ARQUIVOS\MPLAB\EXAMPLES\ID.ASH 41 : Found opcode in column 1. (BSF)
Warning[203] C:\ARQUIVOS\MPLAB\EXAMPLES\ID.ASH 42 : Found opcode in column 1. (goto)
Warning[205] C:\ARQUIVOS\MPLAB\EXAMPLES\ID.ASH 43 : Found directive in column 1. (END)

Build completed successfully.
  
```

## 7 CONCLUSÕES

Os resultados obtidos foram testados no software montador MPLAB da Microchip © e não apresentaram quaisquer erros de sintaxe ou semântica para circuito utilizando portas de entrada e saída e contadores.

O software mostrou-se capaz de gerar código para os dois microcontroladores e também uma ferramenta útil para quem está iniciando na programação destes componentes. A ferramenta abstrai o conhecimento de todos os registradores e detalhes que são gerenciados pelos software. Basta ao usuário conhecer a lógica da programação ladder e configurar algumas variáveis.

A ferramenta também é útil para programadores experientes na linguagem por que pode-se editar o texto gerado na ferramenta e aperfeiçoá-lo.

As dificuldades encontradas no trabalho foram relativas a documentação. Não foram encontrados na biblioteca central livros referentes a programação Ladder, nem referentes ao PIC16F873, porém conseguiu-se obter o material necessário na internet.

Outra dificuldade foi implementar um software dessa complexidade, com tantas classes e métodos e realizar os testes, num espaço tão curto de tempo.

O software atende aos objetivos gerando código para o 8051 e PIC16F873 usando portas de entrada e saída e contadores.

### 7.1 EXTENSÕES

Para continuação deste trabalho sugira que seja feita uma ferramenta simuladora de ambiente ladder, para testar os programas estão semanticamente corretos, antes de serem compilados.

Outra sugestão seria acrescentar microcontroladores a esta lista, editando novos arquivos de configuração.

## REFERÊNCIAS BIBLIOGRÁFICAS

AHO, Alfred V.; SETHI, Ravi; ULLMAN, Jeffrey D. **Compiladores: princípios, técnicas e ferramentas**. Rio de Janeiro: Guanabara Koogan, 1995.

LIMA, Alessandro de Souza. **Microcurso Microcontrolador 8051**, [S.l.], [2002?]. Disponível em: <<http://lula.dmat.furg.br/~vagner/8051emu/apostila/>>. Acesso em: 3 jun. 2003.

MICROCHIP. **Documentação do microcontrolador PIC16F873**, [S.l.], [2002?]. Disponível em: <<http://www.microchip.com>>. Acesso em: 3 fev. 2003.

NICOLOSI, Denys Emilio Campion. **Microcontrolador 8051 detalhado**. São Paulo: Erica, 2000.

SILVA JR, Vidal Pereira da. **Microcontroladores PIC**. São Paulo: Litec, 1998.

TRIANGLE RESEARCH, **Software de simulação Ladder**, San Jose, Estados Unidos, [2002?]. Disponível em: <<http://www.tri-plc.com>>. Acesso em: 8 dez. 2002.



## ANEXO 1 – CONJUNTO DE INSTRUÇÕES DO PIC16F873

ADDWF	f, d	Adiciona W e f
ANDWF	f, d	AND de W com f
CLRF	f	Zera f
CLRW	-	Zera W
COMF	f, d	Complementa f
DECf	f, d	Decrementa f
DECFSZ	f, d	Decrementa f, avança se 0
INCF	f, d	Incrementa f
INCFSZ	f, d	Incrementa f, avança se 0
IORWF	f, d	Ou inclusive entre W e f
MOVF	f, d	Move f
MOVWF	f	Move W para f
NOP	-	Nenhuma operação
RLF	f, d	Rotaciona f para esquerda
RRF	f, d	Rotaciona f para direita
SUBWF	f, d	Subtrai W de f
SWAPF	f, d	Troca nibbles (4 bits) em f
XORWF	f, d	Ou exclusivo entre W e f
BCF	f, b	Zera bit em f
BSF	f, b	Seta bit em f
BTFSC	f, b	Testa bit f, avança se zero
BTFSS	f, b	Testa bit f, avança se um
ADDLW	k	Soma literal e W
ANDLW	k	AND de literal com W
CALL	k	Chama subrotina
CLRWDT	-	Zera Watchdog Timer
GOTO	k	Vai para o endereço
IORLW	k	Ou inclusivo entre literal e W
MOVLW	k	Move literal para W
RETFIE	-	Retorna da interrupção
RETLW	k	Retorna como literal em W
RETURN	-	Retorna de uma subrotina
SLEEP	-	Vai para o modo stand-by
SUBLW	k	Subtrai W de literal
XORLW	k	OU exclusivo entre literal e W

## ANEXO 2 – CONJUNTO DE INSTRUÇÕES DO 8051

### Instruções de transporte

MOV A,Rn	Move o Registro n para o Acumulador
MOV A,Direto	Move o conteúdo da posição de memória para o Acumulador
MOV A,@Ri	Move o conteúdo da RAM interna endereçada po Ri para o Acumulador
(MOV A,#Dado	Move o Dado para o Acumulador
MOV Rn,A	Move o conteúdo do Acumulador para o Registro n
MOV Rn,Direto	Move o conteúdo da memória para o Registro n;
MOV Rn,#Dado	Move o Dado para o Registro n
MOV Direto,A	Move o conteúdo do Acumulador para a posição de memória
MOV Direto,Rn	Move o conteúdo do Registro n para a posição de memória
MOV Direto1,Direto2	Move o conteúdo da posição de memória 2 para a posição de memória 1
MOV Direto,@Ri	Move o conteúdo da posição de memória endereçada por Ri para a posição de memória
MOV Direto,#Dado	Move o Dado para a posição de memória
MOV @Ri,A	Move o conteúdo do Acumulador para a posição de memória endereçada por Ri
MOV @Ri,Direto	Move o conteúdo da posição de memória para a posição de memória endereçada por Ri
MOV @Ri,#Dado	Move o Dado para a posição de memória endereçada por Ri
MOV DPTR,#Dado 16	Move o Dado de 16 bits para o Registro DPTR
MOVC A,@A+DPTR	Move o conteúdo da posição de memória da ROM endereçada por A + DPTR. O endereço será de 16 bits
MOVC A,@A+PC	Move o conteúdo da posição de memória da ROM endereçada por A + PC. O endereço será de 16 bits
MOVX A,@Ri	Move o conteúdo da posição de memória da RAM externa endereçada por Ri para o Acumulador
MOVX A,@DPTR	Move o conteúdo da posição de memória da RAM externa endereçada por DPTR para o Acumulador
MOVX @Ri,A	Move o conteúdo do Acumulador para a posição de memória da RAM externa endereçada por Ri
MOVX @DPTR,A	Move o conteúdo do Acumulador para a posição de memória da RAM externa endereçada por DPTR. (1 byte – 24 pulsos);
PUSH Direto	Coloca na pilha o conteúdo da posição de memória. Incrementa o SP (Stack Pointer) e escreve na pilha
POP Direto	Retira da pilha o Dado e coloca na posição de memória
XCH A,Rn	Troca entre si os conteúdo do Acumulador e do Registro n.
XCH A,Direto	Troca entre si os conteúdo do Acumulador e do Registro n.
XCH A,@Ri	Troca entre si os conteúdo do Acumulador e da posição de memória endereçada por Ri
XCHD A,@Ri	Troca os nibbles menos significativos do conteúdo do Acumulador e da posição de memória endereçada por Ri.

**Instruções para Variáveis Lógicas**

CLR C	Zera o Carry
CLR Bit	Zera o bit Endereçado
SETB C	Seta o Carry
SETB Bit	Seta o bit endereçado
CPL C	Complementa o Carry
CPL Bit	Complementa o bit endereçado
ANL C, Bit	Operação AND entre o Carry e o bit endereçado
ANL C, /Bit	Operação AND entre o Carry e o complemento do bit endereçado
ORL C, Bit	Operação OR entre o Carry e o bit endereçado
ORL C, /Bit	Operação OR entre o Carry e o complemento do bit endereçado
MOV C, Bit	Move o bit endereçado para o Carry
MOV Bit, C	Move o Carry para o bit endereçado
JC rel	Salta se o Carry for "1". O jump é relativo
JNC rel	Salta se o Carry for "0". O jump é relativo
JB Bit, rel	Salta se o bit endereçado estiver em "1"
JNB Bit, rel	Salta se o bit endereçado estiver em "0"
JBC Bit, rel	Salta se o bit endereçado estiver em "1" depois zera o bit

**Instruções Aritméticas**

ADD A, Rn	Soma o conteúdo do Registro n ao Acumulador
ADD A, Direto	Soma o conteúdo da posição de memória ao Acumulador
ADD A, @Ri	Soma o conteúdo da posição de memória endereçada por Ri ao Acumulador
ADD A, #Dado	Soma o Dado ao Acumulador
ADDC A, Rn	Soma o conteúdo do Registro n e o Carry ao Acumulador
ADDC A, Direto	Soma o conteúdo da posição de memória e o Carry ao Acumulador
ADDC A, @Ri	Soma o conteúdo da posição de memória endereçada por Ri e o Carry ao Acumulador
ADDC A, #Dado	Soma o Dado e o Carry ao Acumulador
SUBB A, Rn	Subtrai o conteúdo do Registro n e o Carry do Acumulador
SUBB A, Direto	Subtrai o conteúdo da posição de memória e o Carry do Acumulador
SUBB A, @Ri	Subtrai o conteúdo da posição de memória endereçada por Ri e o Carry do Acumulador
SUBB A, #Dado	Subtrai o Dado e o Carry do Acumulador
INC A	Incrementa o Acumulador
INC Rn	Incrementa o Registro n
INC Direto	Incrementa o conteúdo da posição de memória
INC @Ri	Incrementa o conteúdo da posição de memória endereçada por Ri
DEC A	Decrementa o Acumulador
DEC Rn	Decrementa o Registro n
DEC Direto	Decrementa o conteúdo da posição de memória
DEC @Ri	Decrementa o conteúdo da posição de memória endereçada por Ri
INC DPTR	Incrementa o DPTR
MUL AB	Multiplica A e B. O resultado fica: parte mais significativa em B e menos significativa em Acumulador
DIV AB	Divide A e B. O resultado fica: a parte inteira no Acumulador e o resto em B
DA A	Faz o ajuste decimal do acumulador

**Instruções de desvio**

ACALL End 11	Chama sub-rotina numa faixa de 2 Kbytes da atual posição
LCALL End 16	Chama sub-rotina em qualquer posição da memória de programa (ROM)
RET	Retorno de sub-rotina
RETI	Retorno de rotina de interrupção
AJMP End 11	Salta para outro endereço numa faixa de 2Kbytes da atual
LJMP End 16	Salta para qualquer posição de memória de programa (ROM)
SJMP rel	Salto curto relativo. Salta 127 posições para frente ou 128 para trás
JMP @A,DPTR	Salta para o endereço A + DPTR
JZ rel	Salta se o Acumulador for zero
JNZ rel	Salta se o Acumulador não for zero
CJNE A, Direto, rel	Compara e salta se o Acumulador for diferente da memória endereçada
CJNE A,#Dado,rel	Compara e salta se o Acumulador for diferente do Dado
CJNE Rn,#Dado,rel	Compara e salta se o Registro n for diferente do Dado
CJNE @Ri,#Dado,rel	Compara e salta se o conteúdo da RAM externa endereçada for diferente do Dado
DJNZ Rn,rel	Decrementa o Registro n e salta se for diferente de zero
DJNZ Direto,rel	Decrementa o conteúdo da posição de memória e salta se for diferente de zero
NOP	Nenhuma operação

**Instruções Lógicas**

ANL A,Rn	Executa a operação AND entre o Registro n e o Acumulador
ANL A,Direto	Executa a operação And entre o conteúdo da posição de memória e o Acumulador
ANL A,@Ri	Executa a operação AND entre o conteúdo da posição de memória endereçada por Ri.
ANL A,#Dado	Executa a operação AND entre o Dado e o Acumulador
ANL Direto,A	Executa a operação AND entre o conteúdo da posição endereçada e Acumulador
ANL Direto,#Dado	Executa a operação AND entre a posição de memória endereçada e Dado
ORL A,Rn	Executa a operação OR entre o Registro n e o Acumulador
ORL A,Direto	Executa a operação OR entre o conteúdo da posição de memória e o Acumulador
ORL A,@ Ri	Executa a operação OR entre o conteúdo da posição de memória endereçada por Ri
ORL A,#Dado	Executa a operação OR entre o Dado e o Acumulador. (2 bytes – 12 pulsos);
ORL Direto,A	Executa a operação OR entre o conteúdo da posição endereçada e Acumulador
ORL Direto,#Dado	Executa a operação OR entre a posição de memória endereçada e o Dado
XRL A,Rn	Executa a operação "OU EXCLUSIVO" entre o Registro n e o Acumulador
XRL A,Direto	Executa a operação "OU EXCLUSIVO" entre o conteúdo da posição de memória e o Acumulador. O resultado fica no Acumulador
XRL A,@Ri	Executa a operação "OU EXCLUSIVO" entre o conteúdo da posição de memória endereçada por Ri e o Acumulador
XRL A,#Dado	Executa a operação OU EXCLUSIVO" entre o Dado e o Acumulador
XRL Direto,A	Executa a operação "OU EXCLUSIVO" entre o conteúdo da posição de memória e o Acumulador O resultado fica na posição de memória
XRL Direto,#Dado	Executa a operação "OU EXCLUSIVO" entre o Dado e o conteúdo da posição de memória. O resultado fica na posição de memória
CLR A	Zera o Acumulador
CPL A	Complementa o Acumulador
RL A	Desloca o Acumulador à esquerda
RLC A	Desloca o Acumulador à esquerda através do Carry
RR A	Desloca o Acumulador à direita. (1 byte – 12 pulsos);
RRC A	Desloca o Acumulador à direita através do Carry
SWAP A	Troca o nibble inferior do Acumulador com o superior. Equivale a 4 vezes RR A ou RL A

## ANEXO 3 – ARQUIVO DE CONFIGURAÇÃO DO PIC16F873

```

<Properties>
Name=PIC16F873
<Ports>
//
Pin2=[A,I,O;PORTA;TRISA;0;0;bcf STATUS, RP0;Ra0;]
Pin3=[I,O;PORTA;TRISA;1;1;bcf STATUS, RP0;Ra1;]
Pin4=[I,O;PORTA;TRISA;2;2;bcf STATUS, RP0;Ra2;]
Pin5=[I,O;PORTA;TRISA;3;3;bcf STATUS, RP0;Ra3;]
Pin6=[I,O;PORTA;TRISA;4;4;bcf STATUS, RP0;Ra4;]
Pin7=[I,O;PORTA;TRISA;5;5;bcf STATUS, RP0;Ra5;]
Pin21=[I,O;PORTB;TRISB;0;0;bcf STATUS, RP0;Rb0;]
Pin22=[I,O;PORTB;TRISB;1;1;bcf STATUS, RP0;Rb1;]
Pin23=[I,O;PORTB;TRISB;2;2;bcf STATUS, RP0;Rb2;]
Pin24=[I,C,O;PORTB;Trisb;3;3;bcf STATUS, RP0;Rb3;]
Pin25=[I,C,O;PORTB;Trisb;4;4;bcf STATUS, RP0;Rb4;]
Pin26=[I,C,O;PORTB;Trisb;5;5;bcf STATUS, RP0;Rb5;]
Pin27=[I,C,O;PORTB;Trisb;6;6;bcf STATUS, RP0;Rb6;]
Pin27=[I,C,O;PORTB;Trisb;7;7;bcf STATUS, RP0;Rb7;]
Pin11=[T,I,O;PORTC;Trisc;0;0;bcf STATUS, RP0;Rc0;]
Pin12=[I,O;PORTC;Trisc;1;1;bcf STATUS, RP0;Rc1;]
Pin13=[I,O;PORTC;Trisc;2;2;bcf STATUS, RP0;Rc2;]
Pin14=[I,O;PORTC;Trisc;3;3;bcf STATUS, RP0;Rc3;]
Pin15=[I,O;PORTC;Trisc;4;4;bcf STATUS, RP0;Rc4;]
Pin16=[I,O;PORTC;Trisc;5;5;bcf STATUS, RP0;Rc5;]
Pin17=[I,O;PORTC;Trisc;6;6;bcf STATUS, RP0;Rc6;]
Pin18=[I,O;PORTC;Trisc;7;7;bcf STATUS, RP0;Rc7;]
</Ports>
<Timer>
Timer#8=[INTCON;OPTION_REG;INTCON]
Timer#16=[T1CON;T1CON;T1CON;4]
</Timer>
<Memory>
Mem=[0CH;0;bcf STATUS,RP0]
Mem=[0CH;1;bcf STATUS,RP0]
Mem=[0CH;2;bcf STATUS,RP0]
Mem=[0CH;3;bcf STATUS,RP0]
Mem=[0CH;4;bcf STATUS,RP0]
Mem=[0CH;5;bcf STATUS,RP0]
Mem=[0CH;6;bcf STATUS,RP0]
Mem=[0CH;7;bcf STATUS,RP0]
Mem=[0DH;0;bcf STATUS,RP0]
Mem=[0DH;1;bcf STATUS,RP0]
Mem=[0DH;2;bcf STATUS,RP0]
Mem=[0DH;3;bcf STATUS,RP0]
Mem=[0DH;4;bcf STATUS,RP0]
Mem=[0DH;5;bcf STATUS,RP0]
Mem=[0DH;6;bcf STATUS,RP0]
Mem=[0DH;7;bcf STATUS,RP0]
Mem=[0EH;0;bcf STATUS,RP0]
Mem=[0EH;1;bcf STATUS,RP0]
Mem=[0EH;2;bcf STATUS,RP0]
Mem=[0EH;3;bcf STATUS,RP0]
Mem=[0EH;4;bcf STATUS,RP0]
Mem=[0EH;5;bcf STATUS,RP0]
Mem=[0EH;6;bcf STATUS,RP0]
Mem=[0EH;7;bcf STATUS,RP0]

```

Mem=[0FH;0;bcf STATUS,RP0]  
Mem=[0FH;1;bcf STATUS,RP0]  
Mem=[0FH;2;bcf STATUS,RP0]  
Mem=[0FH;3;bcf STATUS,RP0]  
Mem=[0FH;4;bcf STATUS,RP0]  
Mem=[0FH;5;bcf STATUS,RP0]  
Mem=[0FH;6;bcf STATUS,RP0]  
Mem=[0FH;7;bcf STATUS,RP0]  
Mem=[10H;0;bcf STATUS,RP0]  
Mem=[10H;1;bcf STATUS,RP0]  
Mem=[10H;2;bcf STATUS,RP0]  
Mem=[10H;3;bcf STATUS,RP0]  
Mem=[10H;4;bcf STATUS,RP0]  
Mem=[10H;5;bcf STATUS,RP0]  
Mem=[10H;6;bcf STATUS,RP0]  
Mem=[10H;7;bcf STATUS,RP0]  
Mem=[11H;0;bcf STATUS,RP0]  
Mem=[11H;1;bcf STATUS,RP0]  
Mem=[11H;2;bcf STATUS,RP0]  
Mem=[11H;3;bcf STATUS,RP0]  
Mem=[11H;4;bcf STATUS,RP0]  
Mem=[11H;5;bcf STATUS,RP0]  
Mem=[11H;6;bcf STATUS,RP0]  
Mem=[11H;7;bcf STATUS,RP0]  
Mem=[12H;0;bcf STATUS,RP0]  
Mem=[12H;1;bcf STATUS,RP0]  
Mem=[12;2;bcf STATUS,RP0]  
Mem=[12;3;bcf STATUS,RP0]  
Mem=[12;4;bcf STATUS,RP0]  
Mem=[12;5;bcf STATUS,RP0]  
Mem=[12;6;bcf STATUS,RP0]  
Mem=[12;7;bcf STATUS,RP0]  
Mem=[13;0;bcf STATUS,RP0]  
Mem=[13;1;bcf STATUS,RP0]  
Mem=[13;2;bcf STATUS,RP0]  
Mem=[13;3;bcf STATUS,RP0]  
Mem=[13;4;bcf STATUS,RP0]  
Mem=[13;5;bcf STATUS,RP0]  
Mem=[13;6;bcf STATUS,RP0]  
Mem=[13;7;bcf STATUS,RP0]  
Mem=[14;0;bcf STATUS,RP0]  
Mem=[14;1;bcf STATUS,RP0]  
Mem=[14;2;bcf STATUS,RP0]  
Mem=[14;3;bcf STATUS,RP0]  
Mem=[14;4;bcf STATUS,RP0]  
Mem=[14;5;bcf STATUS,RP0]  
Mem=[14;6;bcf STATUS,RP0]  
Mem=[14;7;bcf STATUS,RP0]  
Mem=[15;0;bcf STATUS,RP0]  
Mem=[15;1;bcf STATUS,RP0]  
Mem=[15;2;bcf STATUS,RP0]  
Mem=[15;3;bcf STATUS,RP0]  
Mem=[15;4;bcf STATUS,RP0]  
Mem=[15;5;bcf STATUS,RP0]  
Mem=[15;6;bcf STATUS,RP0]  
Mem=[15;7;bcf STATUS,RP0]  
Mem=[16;0;bcf STATUS,RP0]  
Mem=[16;1;bcf STATUS,RP0]

Mem=[16;2;bcf STATUS,RP0]  
Mem=[16;3;bcf STATUS,RP0]  
Mem=[16;4;bcf STATUS,RP0]  
Mem=[16;5;bcf STATUS,RP0]  
Mem=[16;6;bcf STATUS,RP0]  
Mem=[16;7;bcf STATUS,RP0]  
Mem=[17;0;bcf STATUS,RP0]  
Mem=[17;1;bcf STATUS,RP0]  
Mem=[17;2;bcf STATUS,RP0]  
Mem=[17;3;bcf STATUS,RP0]  
Mem=[17;4;bcf STATUS,RP0]  
Mem=[17;5;bcf STATUS,RP0]  
Mem=[17;6;bcf STATUS,RP0]  
Mem=[17;7;bcf STATUS,RP0]  
Mem=[18;0;bcf STATUS,RP0]  
Mem=[18;1;bcf STATUS,RP0]  
Mem=[18;2;bcf STATUS,RP0]  
Mem=[18;3;bcf STATUS,RP0]  
Mem=[18;4;bcf STATUS,RP0]  
Mem=[18;5;bcf STATUS,RP0]  
Mem=[18;6;bcf STATUS,RP0]  
Mem=[18;7;bcf STATUS,RP0]  
Mem=[19;0;bcf STATUS,RP0]  
Mem=[19;1;bcf STATUS,RP0]  
Mem=[19;2;bcf STATUS,RP0]  
Mem=[19;3;bcf STATUS,RP0]  
Mem=[19;4;bcf STATUS,RP0]  
Mem=[19;5;bcf STATUS,RP0]  
Mem=[19;6;bcf STATUS,RP0]  
Mem=[19;7;bcf STATUS,RP0]  
Mem=[1A;0;bcf STATUS,RP0]  
Mem=[1A;1;bcf STATUS,RP0]  
Mem=[1A;2;bcf STATUS,RP0]  
Mem=[1A;3;bcf STATUS,RP0]  
Mem=[1A;4;bcf STATUS,RP0]  
Mem=[1A;5;bcf STATUS,RP0]  
Mem=[1A;6;bcf STATUS,RP0]  
Mem=[1A;7;bcf STATUS,RP0]  
</Memory>  
<Counter>  
Mem=[3FH;bcf STATUS,RP0]  
Mem=[40H;bcf STATUS,RP0]  
Mem=[41H;bcf STATUS,RP0]  
Mem=[42H;bcf STATUS,RP0]  
Mem=[43H;bcf STATUS,RP0]  
Mem=[44H;bcf STATUS,RP0]  
Mem=[45H;bcf STATUS,RP0]  
Mem=[46H;bcf STATUS,RP0]  
Mem=[47H;bcf STATUS,RP0]  
Mem=[48H;bcf STATUS,RP0]  
Mem=[49H;bcf STATUS,RP0]  
Mem=[4AH;bcf STATUS,RP0]  
Mem=[4B;bcf STATUS,RP0]  
Mem=[4C;bcf STATUS,RP0]  
Mem=[4D;bcf STATUS,RP0]  
Mem=[4E;bcf STATUS,RP0]  
Mem=[4F;bcf STATUS,RP0]  
Mem=[50;bcf STATUS,RP0]

```

Mem=[51;bcf STATUS,RP0]
Mem=[52;bcf STATUS,RP0]
Mem=[53;bcf STATUS,RP0]
Mem=[54;bcf STATUS,RP0]
Mem=[55;bcf STATUS,RP0]
Mem=[56;bcf STATUS,RP0]
Mem=[57;bcf STATUS,RP0]
Mem=[58;bcf STATUS,RP0]
Mem=[59;bcf STATUS,RP0]
Mem=[5A;bcf STATUS,RP0]
Mem=[5B;bcf STATUS,RP0]
Mem=[5C;bcf STATUS,RP0]
Mem=[5D;bcf STATUS,RP0]
Mem=[5E;bcf STATUS,RP0]
Mem=[5F;bcf STATUS,RP0]
Mem=[60;bcf STATUS,RP0]
</Counter>

```

```
</>
```

```
<Flow>
```

```
</>
```

```
<BNF>
```

```

ASemOr-> BCF &.FLG| BTFSC &1.FLG|BSF &.FLG|BTFSC &2.FLG|BSF &.FLG;
ASemAnd->BSF &.FLG|BTFSS &1.FLG|BCF &.FLG|BTFSS &2.FLG|BCF &.FLG;
ASemNot-> BTFSC &1.FLG| BCF &.FLG| BTFSS &1.FLG| BSF &.FLG;
ASemInstCont->#define CNT&.FLG &.STB;
ASemInstMem->#define &.FLG &.RGC,&.PNC;
ASemInstTimer8->*;
ASemInstTimer16->*;
ASemInstInput->#define &.FLG &.RGC,&.PNC;
ASemInstOutput->bcf &.RGI,&.PNI|bcf &.FLG;
ASemInstAD->bsf &.RGI,&.PNI;
FAD-> *;
FParaTimer->FStopTimer&.NOM:| bcf |RETURN;
FATimer->FGotimer&.NOM:|RETURN;
FRstTimer-> *;
FDecCnt->FDecCnt&.FLG:| BCF STATUS,Z| DECF CNT&.FLG,W| MOVWF
CNT&.FLG| btfsc STATUS,Z| bsf &.FLG|RETURN;
FIncCnt->FIncCnt&.FLG:|BCF STATUS,Z|INCF CNT&.FLG,W|MOVWF CNT&.FLG|XORLW
&.VLR|btfsc STATUS,Z|bsf &.FLG|RETURN;
FRstCnt->FRstCnt&.FLG:|MOVLW 0|MOVWF CNT&.FLG|BCF &.FLG|RETURN ;
FCmpAdMenor-> FCmpAdMenor&.FLG:|bcf &.FLG|bsf ADCON0,0|bsf
ADCON0,2|CAPTUR&.FLG:|BTFSC ADCON0,2|GOTO CAPTUR&.FLG|MOVLW &.ADH|SUBWF
ADRESH,F|BTFSC STATUS,C|RETURN|MOVLW &.ADL|SUBWF ADRESL,F|BTFSC
STATUS,C|RETURN|BSF &.FLG|RETURN;
FCmpAdMaior-> FCmpAdMaior&.FLG:|bcf &.FLG|bsf ADCON0,0|bsf
ADCON0,2|CAPTRA&.FLG:|BTFSC ADCON0,2|GOTO CAPTRA&.FLG|MOVLW &.ADH|SUBWF
ADRESH,F|BTFSS STATUS,C|RETURN|MOVLW &.ADL|SUBWF ADRESL,F|BTFSS
STATUS,C|RETURN|BSF &.FLG|RETURN;
FCmpAdIqual-> FCmpAdIqual&.FLG:|bcf &.FLG|bsf ADCON0,0|bsf
ADCON0,2|CAPTURA&.FLG:|BTFSC ADCON0,2|GOTO CAPTURA&.FLG|MOVLW &.ADH|XORWF
ADRESH,F|BTFSS STATUS,Z|RETURN|MOVLW &.ADL|XORWF ADRESL,F|BTFSS
STATUS,Z|RETURN|BSF &.FLG|RETURN;
MReset-> *;
MSet-> *;
Mem-> *;
Saidas-> *;

```



```

Saida->BTFSS &.FLG|BCF &1.FLG|BTFSC &.FLG|BSF &1.FLG;
SNegada->BTFSS &.FLG|BSF &1.FLG|BTFSC &.FLG|BCF &1.FLG;
SSet->BTFSC &.FLG|BSF &1.FLG;
SReset->BTFSC &.FLG|BCF &1.FLG;
TestaTimer8->BCF STATUS,Z|BTFSC INTCON,T0IF|goto TST1&.NOM|INCF &.FLG1,F|BTFSC
STATUS,Z|INC;
TestaTimer16-> *;
IniciaAd-> MOVLW 2|MOVWF ADCON1|BSF ADCON,7;
TestaContador->*;
TestaSTN-> *;
TestaSTP-> *;
Subrotina->&SUBR;;
VaiPara->goto &SUBR;
Cabecalho->list p=16F873|radix dec|include <P16F873.INC>;
SeVaiPara->BTFSC &.FLG|call &SUBR;
FParaTimer8->FStopTimer&.FLG|RETURN;
FATimer8->FGoTimer&.FLG|RETURN;
FRstTimer8->FRstTimer&.FLG|bcf &.FLG|bcf &.FLG1|CLRF TMR0|MOVLW 130|MOVWF
TMR0|MOVWF &.RG1|MOVLW &.VLR|& MOVWF &.RG2 ;

```

\* Função não implementada

## ANEXO 4 – ARQUIVO DE CONFIGURAÇÃO DO 8051

```

Name=8051
<Ports>
//
Pin1=[I,O;P1;None;0;0;P1.0;]
Pin2=[I,O;P1;None;1;1;P1.1;]
Pin3=[I,O; P1;None;2;2;P1.2;]
Pin4=[I,O; P1;None;3;3; P1.3;]
Pin5=[I,O; P1;None;4;4;P1.4;]
Pin6=[I,O; P1;None;5;5;P1.5;]
Pin7=[I,O; P1;None;6;6;P1.6;]
Pin8=[I,O; P1;None;7;7;P1.7;]
Pin10=[I,O; P3;None;0;0;P3.0;]
Pin11=[I,O; P3;None;1;1;P3.1;]
Pin12=[I,O; P3;None;2;2;P3.2;]
Pin13=[I,O; P3;None;3;3;P3.3;]
Pin14=[I,O; P3;None;4;4;P3.4;]
Pin15=[I,O; P3;None;5;5;P3.5;]
Pin16=[I,O; P3;None;6;6;P3.6;]
Pin17=[I,O; P3;None;7;7;P3.7;]
Pin39=[I,O; P0;None;0;0;P0.0;]
Pin38=[I,O; P0;None;1;1;P0.1;]
Pin37=[I,O; P0;None;2;2;P0.2;]
Pin36=[I,O; P0;None;3;3; P0.3;]
Pin35=[I,O; P0;None;4;4; P0.4;]
Pin34=[I,O; P0;None;5;5; P0.5;]
Pin33=[I,O; P0;None;6;6;P0.6;]
Pin32=[I,O; P0;None;7;7;P0.7;]
Pin28=[I,O; P2;None;0;0;P2.7;]
Pin27=[I,O; P2;None;1;1;P2.6;]
Pin26=[I,O; P2;None;2;2;P2.5;]
Pin25=[I,O; P2;None;3;3;P2.4;]
Pin24=[I,O; P2;None;4;4;P2.3;]
Pin23=[I,O; P2;None;5;5;P2.2;]
Pin22=[I,O; P2;None;6;6;P2.1;]
Pin21=[I,O; P2;None;7;7;P2.0;]
</Ports>
<Timer>
Timer#8= *;
Timer#16=*;
</Timer>
<Memory>
Mem=[00H;;]
Mem=[01H;;]
Mem=[02H;;]
Mem=[03H;;]
Mem=[04H;;]
Mem=[05H;;]
Mem=[06H;;]
Mem=[07H;;]
Mem=[08H;;]
Mem=[09H;;]
Mem=[0AH;;]
Mem=[0BH;;]
Mem=[0CH;;]
Mem=[0DH;;]

```

Mem=[0EH;;]  
Mem=[0FH;;]  
Mem=[10H;;]  
Mem=[11H;;]  
Mem=[12H;;]  
Mem=[13H;;]  
Mem=[14H;;]  
Mem=[15H;;]  
Mem=[16H;;]  
Mem=[17H;;]  
Mem=[18H;;]  
Mem=[19H;;]  
Mem=[1AH;;]  
Mem=[1BH;;]  
Mem=[1CH;;]  
Mem=[1DH;;]  
Mem=[1EH;;]  
Mem=[1FH;;]  
Mem=[20H;;]  
Mem=[21H;;]  
Mem=[22H;;]  
Mem=[23H;;]  
Mem=[24H;;]  
Mem=[25H;;]  
Mem=[26H;;]  
Mem=[27H;;]  
Mem=[28H;;]  
Mem=[29H;;]  
Mem=[2AH;;]  
Mem=[2BH;;]  
Mem=[2CH;;]  
Mem=[2DH;;]  
Mem=[2EH;;]  
Mem=[2FH;;]  
Mem=[30H;;]  
Mem=[31H;;]  
Mem=[32H;;]  
Mem=[33H;;]  
Mem=[34H;;]  
Mem=[35H;;]  
Mem=[36H;;]  
Mem=[37H;;]  
Mem=[38H;;]  
Mem=[39H;;]  
Mem=[3AH;;]  
Mem=[3BH;;]  
Mem=[3CH;;]  
Mem=[3DH;;]  
Mem=[3EH;;]  
Mem=[3FH;;]  
Mem=[40H;;]  
Mem=[41H;;]  
Mem=[42H;;]  
Mem=[43H;;]  
Mem=[44H;;]  
Mem=[45H;;]  
Mem=[46H;;]  
Mem=[47H;;]

```

Mem=[48H;;]
Mem=[49H;;]
Mem=[4AH;;]
Mem=[4BH;;]
Mem=[4CH;;]
Mem=[4DH;;]
Mem=[4EH;;]
Mem=[4FH;;]

```

```
</Memory>
```

```
<Counter>
```

```

Mem=[30H]
Mem=[31H]
Mem=[32H]
Mem=[33H]
Mem=[34H]
Mem=[35H]
Mem=[36H]
Mem=[37H]
Mem=[38H]
Mem=[39H]
Mem=[3AH]
Mem=[3BH]
Mem=[3CH]
Mem=[3DH]
Mem=[3EH]
Mem=[3FH]
Mem=[4FH]
Mem=[50H]
Mem=[51H]
Mem=[52H]
Mem=[53H]
Mem=[54H]
Mem=[55H]
Mem=[56H]
Mem=[57H]
Mem=[58H]
Mem=[59H]
Mem=[5AH]
Mem=[5BH]
Mem=[5CH]
Mem=[5DH]
Mem=[5EH]
Mem=[5FH]
Mem=[60H]
</Counter>

```

```
</>
```

```
<Flow>
```

```
</>
```

```
<BNF>
```

```

ASemOr->MOV C,&1.FLG|ORL C,&2.FLG|MOV &.FLG,C;
ASemAnd->MOV C,&1.FLG|ANL C,&2.FLG|MOV &.FLG,C;
ASemNot->MOV C,&1.FLG|CPL C|MOV &.FLG,C;
ASemInstCont->CNT&.FLG equ &.STB;
ASemInstMem->&.FLG equ &.RGC;
ASemInstTimer8->;

```

```
ASemInstTimer16->;
ASemInstInput->&.FLG equ &.RGC.&.PNC;
ASemInstOutput->CLR &.FLG;
FParaTimer->*;
FATimer->*;
FRstTimer->* ;
FDecCont->*;
FIncCont->*;
FRstCont->*;
MReset->*;
MSet->* ;
Mem->* ;
Saidas->* ;
Saida->MOV C,&.FLG|MOV &1.FLG,C;
SNegada->MOV C,&.FLG|CPL C|MOV &1.FLG,C;
SSet->MOV C,&.FLG;
SReset->MOV C,&.FLG;
TestaTimer16->*;
TestaContador->*;
TestaSTN->* ;
TestaSTP->* ;
Subrotina->&SUBR;;
VaiPara->ljmp &SUBR;
Cabecalho->;
SeVaiPara->JNB &.FLG,1|LCall &SUBR;
```

\* Funções não implementadas.