CENTRO UNIVERSITÁRIO UNIVATES
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
CURSO DE ENGENHARIA DA COMPUTAÇÃO

THIAGO SCHIMUNECK

AMBIENTE DE DESENVOLVIMENTO INTEGRADO PARA ENSINO E PROGRAMAÇÃO DE MICROCONTROLADORES DA FAMÍLIA MCS51

Lajeado
2013

THIAGO SCHIMUNECK

AMBIENTE DE DESENVOLVIMENTO INTEGRADO PARA ENSINO E PROGRAMAÇÃO DE MICROCONTROLADORES DA FAMÍLIA MCS51

Trabalho de Conclusão de Curso apresentado ao Centro de Ciências Exatas e Tecnológicas do Centro Universitário UNIVATES, como parte dos requisitos para a obtenção do título de bacharel em Engenharia da Computação.
Área de concentração: Microcontroladores

ORIENTADOR: Ronaldo Hüsemann

Lajeado

2013

THIAGO SCHIMUNECK

AMBIENTE DE DESENVOLVIMENTO INTEGRADO PARA FAMÍLIA MCS51:

Este trabalho foi julgado adequado para a obtenção do título de bacharel em Engenharia de Computação do CETEC e aprovado em sua forma final pelo Orientador e pela Banca Examinadora.

Orientador: _____

Prof. Ronaldo Hüsemann, UNIVATES

Doutor pelo PPGEE / UFRGS – Porto Alegre, Brasil

Banca Examinadora:

Prof. Ronaldo Hüsemann, UNIVATES

Doutor pelo PPGEE/UFRGS – Porto Alegre, Brasil

Prof. Marcelo de Gomensoro Malheiros, UNIVATES

Mestre pela Engenharia Elétrica/UNICAMP – Campinas, Brasil

Prof. Paulo Roberto Mallmann, UNIVATES

Mestre em Computação Aplicada pela UNISINOS – São Leopoldo, Brasil

Coordenador do Curso de Engenharia da Computação

Prof. Marcelo de Gomensoro Malheiros

Lajeado, junho de 2013.

RESUMO

Este documento descreve o desenvolvimento de um ambiente integrado para programação e depuração em tempo real de softwares escritos para a plataforma Intel MCS51. A solução desenvolvida utiliza ferramentas livres para gerar um aplicativo com versões para MS Windows e Linux permitindo interface para comunicação com uma placa de desenvolvimento externa, onde os programas podem ser executados diretamente. Como recursos de depuração inclui-se execução passo a passo, *breakpoints*, acesso a registradores internos, memória do sistema e variáveis definidas pelo usuário. Como base para o desenvolvimento da ferramenta utilizou-se o compilador livre SDCC, o software monitor CMON51 e bibliotecas gráficas Java.

Palavras-chave: MCS51, depuração, compilação, IDE.

ABSTRACT

This document describe design of an integrated development environment for software real-time programming and debugging for Intel MCS51 architecture. The solution adopt open source tools in order to generate an application with MS Windows and Linux versions, with interactive interface for external development board, where programs can be directly executed. As debugging features it includes step-by-step execution, breakpoints, internal registers, system memory and user-defined variables access. The proposed tool had been used then open source SDCC compiler, CMON51 software monitor and Java graphics libraries.

Keywords: MCS51, debugging, compilation, IDE.

LISTA DE FIGURAS

LISTA DE CÓDIGOS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

ALU:	Arithmetic Logic Unit
ASCII:	American Standard Code for Information Interchange
CMOS:	Complementary Metal-Oxide Semicondutor
E/S:	Entrada/Saída
IDE:	Integrated Development Environment
RAM:	Random Access Memory
ROM:	Read-only Memory
SDCC:	Small Device C Compiler
SFR:	Special Funcion Registers
TTL:	Transistor to Transistor Logic
UART:	Universal Asynchronous Receiver-Transmitter

SUMÁRIO

1 INTRODUÇÃO

O desenvolvimento de software para equipamentos embarcados, mesmo seguindo métodos e especificações, nem sempre é uma tarefa fácil. Muitas vezes depende-se de diferentes ferramentas para construir, compilar, e analisar o código escrito em busca de erros de implementação, falhas de sincronia, problemas alocação de memória, erros de cálculos entre outros problemas possíveis (GIMENEZ, 2005).

Para ambientes acadêmicos, onde se tem disciplinas para o ensino de programação de microcontroladores, ainda é muito utilizada a arquitetura 8051 como base, devido sua facilidade de programação, popularidade e simplicidade (GIMENEZ, 2005).

Um ambiente onde tem-se todas estas ferramentas disponíveis e integradas (IDE) como um único software é algo que gera agilidade no desenvolvimento deste tipo de aplicação. Porém hoje são poucos estes ambientes disponíveis para a plataforma 8051 e os que foram encontrados, na maioria, são pagos e não portáveis a outros sistemas operacionais, como o Linux, por exemplo. Existem disponíveis versões de demonstração, mas com limitações de funcionalidades e tamanho de código compilado (GIMENEZ, 2005).

A UNIVATES utiliza o software chamado KID51 baseado em plataforma MS-DOS e que exige do usuário a conversão de formato do programa compilado e o envio deste de forma manual através de comandos digitados. Tal tarefa torna-se repetitiva e dispendiosa durante o desenvolvimento e depuração dos códigos. Visando a melhoria deste processo, uma ferramenta IDE foi criada para substituir este software.

Uma solução disponível que apresenta um ambiente de desenvolvimento completo é o uVision da empresa Keil. Porém esta solução é proprietária, impondo limites de tamanho de código de 2KBytes em sua versão de demonstração e tem seu funcionamento restrito ao sistema operacional Microsoft Windows.

Outra solução encontrada, esta de código aberto é o aplicativo MCU 8051 escrito em linguagem Tcl/Tk[1] que possui um ambiente integrado com suporte ao SDCC e ao ASM-51, porém apenas permitindo a simulação do programa escrito (LUCAS, 2012).

[1]Tcl/Tk (Tool Command Language) é uma linguagem de código aberto com componentes gráficos bastante popular. (MONTEIRO, 2001)

Considerando este cenário se propõe o desenvolvimento de uma ferramenta que sirva como ambiente integrado de desenvolvimento para a arquitetura 8051, onde estará disponível o processo de edição, compilação e depuração do código sendo este emulado diretamente sobre a placa de desenvolvimento.

Esta ferramenta será desenvolvida integrando-se dois softwares livres disponíveis: o primeiro trata-se do compilador SDCC (Small Device C Compiler), sendo este um compilador de linguagem C de código aberto para microcontroladores da família MCS51, Dallas DS80C390, Motorola HC08, Zilog Z80 e Microchip PIC16 e 18 (ainda não completo) (NICOLOSI; BRONZERI, 2005).

E o segundo trata-se do CMON51, um programa monitor desenvolvido em linguagem C para o SDCC e escrito por Jesus Calvino-Fraga (FRAGA, 2005).

Este programa monitor fará o papel de intermediário entre a emulação do software escrito em linguagem C e o usuário. Para a interação desta será desenvolvida uma interface gráfica em Java a fim de proporcionar recursos básicos para edição dos códigos fontes e demais operações.

Para o experimento de emulação do software sobre um hardware real confeccionou-se uma placa contendo um microcontrolador Atmel AT89S52, sendo este compatível com o 8052, um *latch* 74HC537 para o barramento de endereços, uma memória de 32768 bytes para armazenamento do programa a ser emulado e uma porta lógica 74HC08 para multiplexação dos sinas de leitura de dados e instruções.

Também nesta placa foram adicionados 8 leds ligados à porta P1 a fim de permitir visualização externa de ações do programa em execução. Desta forma pode-se acompanhar a depuração de programas exemplo executados na placa experimental e verificar seu estado atual no ambiente de depuração.

A estrutura desta placa torna-se compatível com os kits de desenvolvimento hoje utilizados pela UNIVATES. Tais kits possuem disponíveis vários periféricos como teclado numérico, display alfanumérico, conversor analógico-digital e conversor digital-analógico. todos mapeados em memória externa de forma a não interferir no funcionamento básico do monitor CMON51.

O Capítulo 2 aborda características do microcontrolador 8051, como memórias, dispositivos de comunicação externa e demais periféricos, enquanto o Capítulo 3 aborda detalhes sobre o compilador SDCC. Já o Capítulo 4 aborda detalhes sobre o software monitor CMON51. Na sequência, o Capítulo 5 aborda soluções já existentes voltadas ao problema

citado. O Capítulo 6 descreve a solução desenvolvida. O Capítulo 7 aborda detalhes sobre o experimento realizado. Por fim, o Capítulo 8 apresenta as considerações finais do autor.

2 A FAMÍLIA MCS51

Nos anos 80 a INTEL lança o sucessor do microcontrolador 8048 de 4 bits, o 8051 de 8 bits (NICOLOSI, 2002). Este veio a ser então o mais popular microcontrolador do mundo, tendo uma vasta família no mercado e sendo produzido por mais de 30 empresas e com mais de 600 variações de modelos (NICOLOSI; BRONZERI, 2005).

Diversas versões do chip foram introduzidas no mercado, como o 8751 com EPROM interna programável pelo usuário, o 8031 que necessita de memória de código externa para funcionamento, o 8052 que possui um temporizador de 16 bits adicional entre outras variações que formaram a família MCS51 (NICOLOSI; BRONZERI, 2005).

2.1 Características principais

As principais características da família MCS51 na sua forma original são:

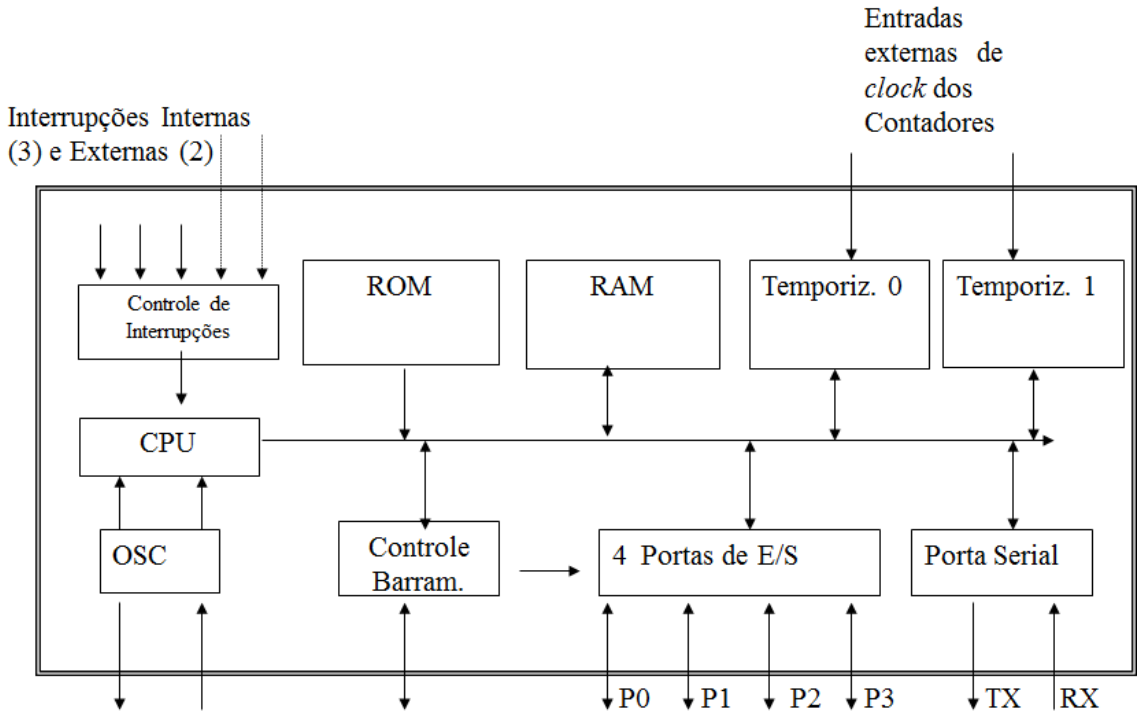
- Memória RAM interna de uso geral de 128 bytes e 128 bytes de registradores especiais;
- 4 Kbytes de ROM interna para código;
- 4 Portas de I/O;
- 2 Timers de 16 bits;
- Endereçamento de ROM externa com capacidade de 64 Kbytes;
- Endereçamento de RAM externa com capacidade de 64 Kbytes;
- Instruções para operação booleana;
- Instrução para divisão e multiplicação direta;
- Entradas de interrupções externas;

A capacidade de endereçamento em 64 Kbytes para RAM e ROM se dá através de um barramento de 16 bits formados pelas portas P0 e P2 como será descrito posteriormente.

2.1.1 Arquitetura interna

A estrutura interna do microcontrolador 8051 está representada na imagem a seguir:

Figura 1 - Arquitetura interna do 8051/8052



Fonte: GIMENEZ (2005).

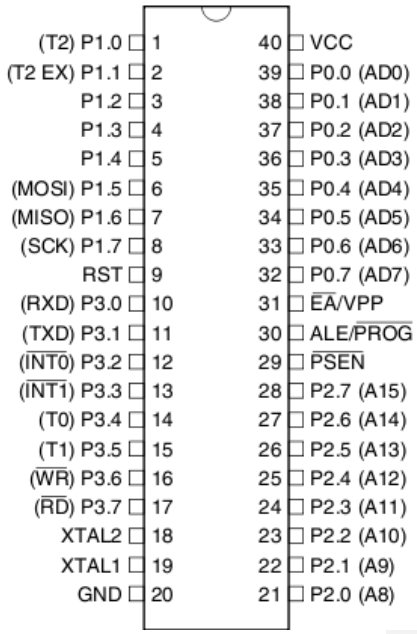
Na Figura 1 verifica-se os principais componentes do 8051, sendo portas P0, P1, P2 e P3 que são destinadas à comunicação externa, RAM interna, temporizadores, interrupções, ROM interna (exceto 8031 e 8032), ALU e demais registradores. (NICOLOSI, 2000).

Nota-se que estão disponíveis usuário 32 pinos de I/O sendo 8 em cada porta, porém estes são compartilhados com demais funções do microcontrolador. A porta P0, assim como a P2, fica comprometida caso seja utilizada memória externa ao chip, sendo que a parte baixa do endereço e o barramento de dados é multiplexado nesta porta assim como a porta P2 que conterá a parte alta do endereço. A porta P3 contém funções alocadas para diversos periféricos como a UART (serial), temporizadores/contadores, interrupções e controle da memória externa. A porta P1 fica livre para utilização como E/S (entrada/saída) exceto na versão que contém um temporizador adicional (8052) ou conversor AD interno.

2.1.2 Esquema elétrico

O aspecto físico do 8051 pode ser visto na Figura 2:

Figura 2 - Pinos do 8051 no encapsulamento PDIP



Fonte: Atmel (2008).

De forma breve descreve-se abaixo funções dos pinos presentes no 8051.

Nos pinos 1 a 8, está alocada a porta P1, do qual trata-se como uma porta I/O de uso geral, sendo que, no caso do 8052, os pinos 1 e 2 controlam funções do temporizador adicional de 16 bits (ATMEL, 2008).

No pino 9 tem-se o reset do microcontrolador, onde pode-se controlar a inicialização do programa no vetor 0x0000. Mais adiante tem-se maiores informações da utilização deste pino.

Nos pinos 10 a 17 tem-se a porta P3, do qual possui diversas aplicações. O pino 10 (P3.0) é responsável pela recepção de dados no caso de utilização da UART interna. O pino 11 é responsável pela transmissão de dados da mesma forma que o anterior. O pino 12 é a entrada para acionamento da primeira interrupção externa. O pino 13 é entrada para o acionamento da segunda interrupção externa. O pino 14 controla funções do temporizador/contador 0 e em sequencia o pino 15 controla funções do temporizador/contador 1. Os pinos 16 e 17 executam funções de escrita e leitura (respectivamente) nas memórias externas.

Nos pinos 18 e 19 tem-se conexão para o cristal oscilador.

No pino 20 tem-se o referencial (ou terra).

Do pino 21 ao pino 28 tem-se a porta P2 do qual pode ser utilizada como I/O de propósito geral ou para endereçamento de memórias externas.

O pino 29 denominado PSEN controla a leitura de informações da memória ROM externa caso esta esteja presente.

O pino 30 denominado ALE controla o latch de multiplexação de dados na porta P0 de forma permitir separação de endereços e dados.

O pino 31 denominado EA permite que seja desabilitado a leitura da ROM interna ao microcontrolador quando este a possui.

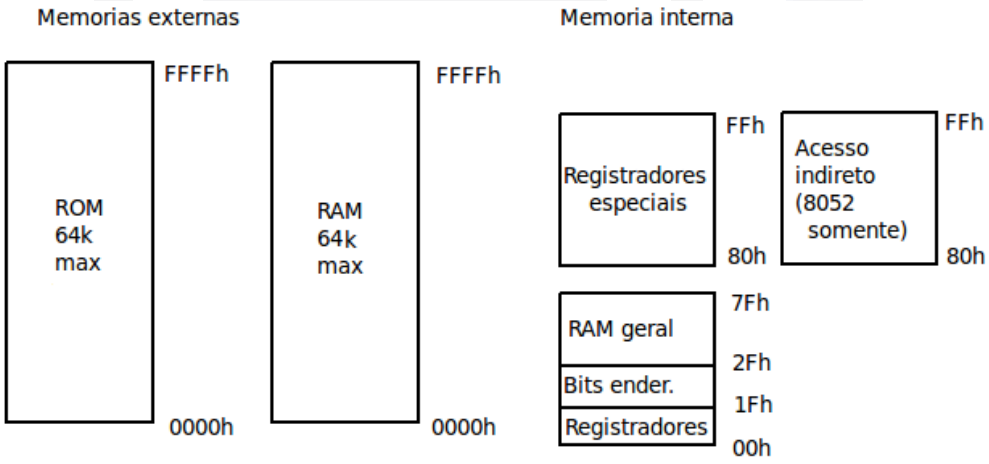
Do pino 32 a 39 tem-se a porta P0 que, como já descrito anteriormente, pode ser utilizado como porta de I/O de propósito geral, ou também de barramento de dados e endereços (parte menos significativa) para o acesso à memórias externas. multiplexa dados e endereços.

O pino 40 é alimentação +5 volts (NICOLOSI, 2000).

2.2 Estrutura de memória

Para este microcontrolador existem três regiões diferentes de memória (GIMENEZ, 2005). A Figura 3 demostra esta organização.

Figura 3 - Diagrama representando memórias do 8051



Fonte: Adaptado de (NICOLOSI; BRONZERI, 2005).

A primeira é a memória de código, que pode estar localizada internamente ao chip ou em um componente externo. É nesta memória que fica alocado o código do programa a ser executado. Do ponto de vista do microcontrolador, esta memória é somente para leitura,

podendo apenas ser consultado valores de seu conteúdo. Além de código de instruções, esta memória também pode armazenar tabelas de dados e textos que podem ser utilizados pelo programa a ser executado. Esta memória pode ser acessada pelo programa através da instrução de leitura de memória de código com o auxílio do registrador DPTR, o qual será descrito mais adiante no texto (NICOLOSI; BRONZERI, 2005).

Geralmente o microcontrolador possui internamente uma certa quantidade de memória código para armazenamento do programa a ser executado. O que permite a utilização do chip sem a necessidade de memória de código externa. A quantidade de memória disponível internamente varia de acordo com o modelo e fabricante. Um exemplo é o AT89S52 que possui 8Kbytes de código interna. Outro fato importante a se observar é que caso o programa faça acesso a um endereço acima do espaço disponível na memória de código interna (2000h no caso do modelo citado) a busca será realizada na memória externa utilizando as portas P0 e P2 e sinal do pino PSEN. Caso esta memória externa não exista, o programa irá perder-se (ATMEL, 2008).

Na sequência tem-se a memória interna com 128 bytes para o 8051 e 256 bytes para o 8052. Esta memória é dividida em três blocos básicos. A Figura 4 demostra a estrutura destes blocos.

Figura 4 - Organização da memória interna do 8051

<table>
<tr><th>Endereços de byte</th><th colspan="2">Região de memória</th></tr>
<tr><td>FF
Registradores (SFR)
Alguns bits endereçáveis
80</td><td>Registradores especiais
(Special Funcion Registers)</td><td>RAM de propósito geral
Apenas acessível indiretamente
MOV A, @R0
Presente apenas no 8052</td></tr>
<tr><td>7F
“Bit não endereçável”
30</td><td rowspan="2">RAM de propósito geral</td><td rowspan="6"></td></tr>
<tr><td>2F
“Bit endereçável”
20</td></tr>
<tr><td>1F
18</td><td>Banco 3</td></tr>
<tr><td>17
10</td><td>Banco 2</td></tr>
<tr><td>0F
08</td><td>Banco 1</td></tr>
<tr><td>07
00</td><td>Banco padrão registradores R0-R7</td></tr>
</table>

Fonte: Adaptado de (NICOLOSI, 2002).

O primeiro bloco é o de endereço 00h até 7Fh, onde se localizam os quatro bancos de registradores de uso geral R0 a R7, a região de memória bit endereçável compreendida entre 20h e 2Fh, e restante de espaço de propósito geral compreendido entre 30h e 7Fh.

No 8052, a partir do endereço 7Fh, existe uma divisão entre área de memória acessível diretamente no qual acessa a SFR (*Special Funcion Registers*) do microcontrolador e a área que acesso indireto sendo que nesta última o usuário tem os 128 bytes.

Algumas instruções do 8051 foram projetadas para aceitar apenas um bit de informação como parâmetro. Um exemplo é a instrução SETB que ajusta o valor de determinado bit para 1. Neste caso, o acesso é feito com endereços de 00h a FFh para acesso a bit, sendo que o endereço 00h representa o bit menos significativo do byte de memória localizado no endereço 20h. Para a região de memória interna baixa (00h a 7Fh) os endereços de bits são sequenciais totalizando 128 bits manipuláveis. Para a região alta da memória interna (80h a FFh) os endereços de bits estão distribuídos entre os diversos SFR (GIMENEZ, 2005).

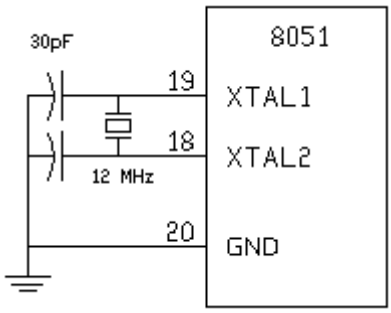
Por último tem-se a memória RAM externa onde pode-se armazenar informações sobre o processamento do programa ou até mesmo código a ser executado pelo microcontrolador desde que seja adaptada uma arquitetura *Von Neumann* a permitir a busca de instruções mesmo na memória RAM (OLIVEIRA; ANDRADE, 2006).

Para acessar a memória RAM utiliza-se da instrução MOVX, indicando que a leitura ou armazenamento da informação deve ocorrer na memória RAM externa. (SILVA JÚNIOR, 2003).

2.3 O Clock e o Reset

Já mencionado anteriormente, o 8051 possui dois pinos para a entrada de sinal de clock. Internamente também há um circuito oscilador que depende apenas de um cristal e capacitores para o funcionamento. O clock regula a velocidade de execução do programa pelo microcontrolador, sendo seu relógio. A frequência mínima de oscilação é de 3,5Mhz e o máximo depende do modelo do chip (NICOLOSI, 2000). A Figura 5 mostra a conexão de um cristal de 12MHz junto ao microcontrolador.

Figura 5 - Esquema de ligação do cristal oscilador



Fonte: Adaptado de (NICOLOSI, 2000).

Também é possível a conexão de uma fonte geradora de relógio externa TTL no qual o pino 19 (XTAL 1) é conectado ao terra e o pino 18 (XTAL 2) recebe o sinal externo (SILVA JR, 2003).

O relógio gera os ciclos de máquina, no qual permite a execução das instruções no chip. Para o 8051 a cada 12 ciclos de relógio gera-se um ciclo de máquina, ou seja, para uma frequência de 12MHz o microcontrolador executa instruções a uma velocidade de 1Mhz (NICOLOSI, 2002).

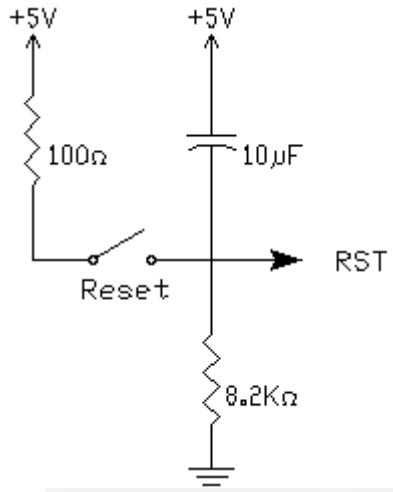
O processo *reset* do 8051 ocorre ao forçar o pino 9 a nível alto durante 2 ou mais ciclos de relógio. Após esta ocorrência o microcontrolador interrompe a sua execução e efetua os seguintes passos:

- os registradores PC, ACC B, Flags, DPTR e demais registradores dos temporizadores são zerados;
- no registrador SP é escrito o valor 07h
- todas as portas de I/O tem sua saída alterada para 0FFh;
- o registrador de controle da UART será zerado e o *buffer* terá valor indeterminado;
- o registrador PCON terá apenas o seu bit mais significativo zerado;
- os registradores IE e IP terão seus valores alterados para XXX00000.

A RAM interna não é alterada pelo *reset*, porém após energizar o chip, seus valores são indeterminados (SILVA JÚNIOR, 2003).

O circuito de *reset* é apresentado na Figura 6.

Figura 6 - Circuito de *reset* no 8051



Fonte: Adaptado de (NICOLOSI; BRONZERI, 2005).

2.4 Interrupções

Segundo Silva Júnior (2003), as interrupções constituem uma ferramenta muito importante nos sistemas microprocessados, pois define uma forma de interromper a execução de um programa através de um evento externo ou interno para dar devida atenção a este.

Sua vantagem está na simplificação do hardware pois dispensa o programa de estar monitorando certos dispositivos periféricos para verificar se estes precisam de atenção.

Nicolosi (2002) afirma que na arquitetura do 8051 as interrupções são do tipo **não vetoradas**[2] o que acaba definindo endereços fixos na memória de código para onde o fluxo de execução é desviado após ocorrência da interrupção.

Na família 8051 são definidas cinco interrupções sendo estas definidas na Tabela 1.

Tabela 1 - Interrupções do 8051/8052

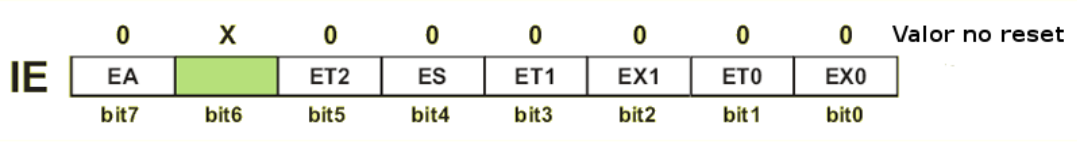
Interrupção	Tipo	Endereço de desvio	Chip
RESET	Externa – RST	0000h	8051
INT0	Externa – Pino P3.2	0003h	8051
INT1	Externa – Pino 3.3	0013h	8051
T/C 0	Interna	000Bh	8051
T/C 1	Interna	001Bh	8051
SERIAL	Interna	0023h	8051
T/C 2	Interna	002Bh	8051

Fonte: Adaptado de (NICOLOSI, 2002).

[2]Interrupções do tipo não vetoradas tem seus endereços de memória atendimento fixos. Para interrupções vetoradas o endereço pode ser alterado pelo programador. (NICOLOSI, 2002)

A habilitação das interrupções é controlada pelo registrador IE cuja estrutura pode ser vista na Figura 7:

Figura 7 - Registrador IE



Fonte: Adaptado de (SILVA JÚNIOR, 2003).

Cada um dos bits permite que se ative determinada interrupção de determinado periférico, sendo que o bit mais significativo (EA) permite ativar e desativar todas as interrupções do microcontrolador de uma só vez, sendo assim uma chave geral para mascarar todas as interrupções.

Segundo Silva Júnior (2003) o registrador IP (*interrupt priority*) permite acoplar cada interrupção a duas filas de atendimento distintas, sendo uma de maior prioridade e outra de menor prioridade. Desta forma é possível até inverter a ordem natural de atendimento das interrupções do 8051.

Para as interrupções externas existem quatro bits definidos no registrador TCON que permitem selecionar o tipo de disparo destas, sendo possível selecionar acionamento por nível ou por borda (GIMENEZ, 2002).

O atendimento de um evento de interrupção sempre ocorre com uma chamada ao endereço do respectivo evento a ser atendido. Neste caso o microcontrolador termina a última instrução que está sendo executada, empilha dois valores na pilha sendo o primeiro o valor da parte baixa do registrador PC e o segundo a parte alta deste mesmo registrador e realiza um desvio para o endereço de atendimento da interrupção (GIMENEZ, 2005).

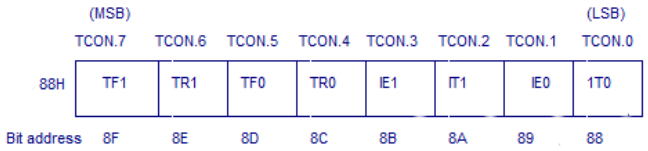
2.5 Temporizadores/Contadores

Os temporizadores/contadores são periféricos que permitem uma contagem baseada na frequência do oscilador do microcontrolador (temporizador) ou de uma fonte externa de pulsos (contador). Seu objetivo é contar de forma precisa o tempo ou determinado número de ocorrências de um evento. O 8051 possui dois temporizadores/contadores disponíveis (no caso do 8032/8052 são três) cujo funcionamento e controle é feito via software e também por pinos externos sendo que cada temporizador/contador é uma unidade autônoma em relação ao restante do microcontrolador (NICOLOSI, 2002).

Cada temporizador/contador possui quatro modos de operação diferentes e permite uma contagem máxima de 65535 (FFFFh) quando configurado para 16 bits. O *overflow* (estouro da contagem) ocorre sempre que o contador atingir o valor máximo e transitar para o valor mínimo novamente, pois sempre a contagem é em sentido crescente. Cada temporizador/contador possui dois registradores sendo TLx e o THx onde ficam armazenados os valores da contagem, sendo estes acessíveis para leitura e escrita via software (SILVA JÚNIOR, 2003).

Para o controle destes dispositivos estão disponíveis dois registradores denominados TCON e TMOD no qual é possível realizar a programação funcionamento e disparo de cada temporizador/contador. A Figura 8 apresenta a estrutura do registrador TCON.

Figura 8 - Registrador TCON



Fonte: Adaptado de (SILVA JÚNIOR, 2003).

No registrador TCON apenas os 4 bits mais significativos são utilizados para configuração dos temporizadores/contadores, sendo que os demais bits definem propriedades específicas das interrupções externas já definidas anteriormente (SILVA JÚNIOR, 2003).

Os bits TR0 e TR1 são setados via software para ativar respectivamente os temporizadores/contadores 0 e T/C 1 e resetados também via software para desativá-los.

Os bits TF0 e TF1 são setados via hardware quando ocorre estouro dos respectivos temporizadores/contadores gerando a solicitação da interrupção. Após o atendimento da interrupção (RETI), este bit é zerado também pelo hardware automaticamente.

A Tabela 2 apresenta o resumo das configurações possíveis para os temporizadores/contadores.

Tabela 2 - Modos de operação dos temporizadores/contadores do 8051

Modo	M1	M0	Descrição
0	0	0	Contador de 13 bits compatível com 8048
1	0	1	Contador de 16 bits
2	1	0	Contador de 8 bits com recarga automática
3	1	1	Temporizador misto

Fonte: Adaptado de (NICOLOSI, 2002).

No modo 0 o temporizador/contador opera utilizando o seu respectivo registrador TH.x para a contagem de 8 bits e o registrador TL.x para uma divisão de frequência de até 32 vezes. Neste caso apenas os 5 bits menos significativos do registrador são utilizados. Neste modo a contagem pode ser no máximo de 8162 vezes (32 x 255).

No modo 1 o temporizador/contador opera na contagem de 16 bits utilizando integralmente os seus registradores TH e TL o que proporciona uma contagem de até 65535.

No modo 2 o temporizador/contador permite uma contagem de 8 bits (até 255) utilizando o registrador TL e, após o estouro da contagem, faz a recarga deste registrador com o valor armazenado no respectivo registrador TH.

No modo 3 o temporizador/contador 1 é desativado e os seus bits de controle passam a operar um novo temporizador/contador de 8 bits existente no registrador TH0. O temporizador/contador 0 opera também em 8 bits sobre o registrador TL0 e tem seu controle pelos seus respectivos bits.

2.6 UART – Canal serial

A família MCS51 possui um canal serial incorporado do tipo Full-Duplex (permite transmissão e recepção ao mesmo tempo) com quatro modos possíveis de operação. Dois registradores são utilizados para operar o canal serial, sendo o SCON para configuração e o SBUF para transmissão e recepção de dados. O registrador SBUF é dividido internamente em dois registradores, sendo que operações de escrita neste registrador afetam o registrador interno de transmissão e operações de leitura recebem dados do registrador interno de recepção (GIMENEZ, 2005).

A Figura 9 apresenta a estrutura do registrador SCON.

Figura 9 - Registrador SCON

7	6	5	4	3	2	1	0
SM0	SM1	SM2	REN	TB8	RB8	TI	RI

Fonte: Adaptado de (NICOLOSI, 2002).

Os bits SM0 e SM1 permitem a programação de quatro modos de operação que são resumidos na Tabela 3. O bit SM2 é utilizado na comunicação entre vários microcontroladores. O bit REN permite habilitar a recepção de dados pela UART. Os bits TB8 e RB8 permitem a escrita ou leitura do nono bit a cada byte transmitido. Para os modos 0

e 1 eles representam o *stop bit* e para os modos 2 e 3 representam o bit de paridade. Os bits TI e RI indicam fim de transmissão e recepção respectivamente e indicam a geração de interrupção da porta serial quando esta está habilitada. Estes bits são setados via hardware mas devem sempre ser resetados via software após transmissão ou recepção sendo esta ou não na rotina de interrupção da serial.

2.6.1 Modos de operação da UART

Como foi mencionado anteriormente, a UART do 8051 possui quatro modos de operação, os quais são resumidos na Tabela 3 e brevemente descritos na sequência.

Tabela 3 - Modos de operação da UART do 8051

Modo	SM0	SM1	Descrição
0	0	0	Fclock / 12
1	0	1	Variável
2	1	0	Fclock / 12 ou Fclock / 64
3	1	1	Variável

Fonte: Adaptado de Silva Junior (2003).

No modo 0 a comunicação ocorre de forma síncrona, onde o pino RXD transmite ou recebe os dados e o pino TXD gera ou recebe o sinal de relógio para o sincronismo. Neste caso a frequência de operação é a mesma do ciclo de máquina (Cristal / 12) o que impõe uma velocidade bem alta na comunicação de dados (SILVA JÚNIOR, 2002).

No modo 1 a comunicação é assíncrona e os pinos TXD e RXD são para transmissão e recepção respectivamente, a frequência é variável podendo ser gerada pelo T/C 1 ou pelo T/C2 (somente no 8052) e são transmitidos 10 bits por byte enviado, sendo 1 start bit, 8 bits de dados e 1 stop bit.

No modo 2 a comunicação é assíncrona, a frequência é selecionável para 1/32 ou 1/64 da frequência do cristal e o pacote de dados é formado por 11 bits sendo estes 1 start bit, 8 bits de dados, 1 bit selecionável através dos bits TB8 e RB8 do registrador SCON e um stop bit.

No modo 3 a comunicação é idêntica ao modo 2 exceto pelo fato de que a frequência de transmissão/recepção é dada pelo T/C1 ou pelo T/C2 (somente 8052).

O registrador SMOD disponibiliza o bit SMOD que quando setado permite dobrar a taxa de transmissão/recepção do canal UART.

2.7 Instruções Assembly

Segundo Silva Júnior. (2003) a família MCS51 é capaz de executar um total de 256 instruções distintas, sendo que estas muitas vezes possuem sua sintaxe (mnemônico) bastante similar estre si, sendo tratada como uma mesma instrução com parâmetros distintos. O tamanho de cada instrução armazenada na memória de código pode variar de 1 byte a 3 bytes, dependendo do número de parâmetros necessários para a execução desta instrução. Cada instrução possui um modo de endereçamento próprio, sendo que é possível classificar cada um destes tipos de endereçamento (NICOLOSI, 2002).

2.7.1 Modos de endereçamento das instruções

Nicolosi (2002) descreve oito modos de endereçamento utilizados pelas instruções do 8051. São eles:

1. Registrador
2. Direto
3. Indireto
4. Imediato
5. Relativo
6. Absoluto
7. Longo
8. Indexado

No modo Registrador (1) as instruções são codificadas de forma haver um código para cada registrador. Isto permite que a instrução saiba com qual dos registradores operará e mesmo assim ocupe apenas 1 byte na memória de código.

Exemplo de instrução: MOV R1, A

No modo Direto (2) as instruções referem-se a um endereço fixo de memória interna fazendo com que ocupem 2 bytes geralmente. Este endereço é definido por uma posição de memória direta ou por seu mnemônico.

Exemplo de instrução: MOV 0x90, A ; 0x90 endereço fixo da porta P1

No modo Indireto (3) as instruções acessam a memória através de ponteiros, ou seja, valores de endereços escritos em outro registrador. Nestes casos os registradores R0, R1 e DPTR podem ser utilizados para indicar o endereço de memória a ser acessado. Este tipo de endereçamento é caracterizado pelo caractere @ antecedendo o nome do registrador.

Exemplo de instrução: MOV A, @R1

No modo Imediato (4) as instruções carregam como parâmetro um dado fixo que será escrito no registrador em questão. É caracterizada pelo carácter # antecedendo o valor a ser escrito no registrador indicado.

Exemplo de instrução: MOV A, #0x20 ; carrega registrador A com o valor 20h

No modo Relativo o endereçamento é definido pelo endereço atual da instrução somado de um deslocamento para o novo endereço. Este valor é um número de 8 bits com sinal variando de -128 a +127 que permite saltos no código somando-se este número a posição da instrução seguinte.

Exemplo de instrução: SJMP 0x8003

No modo Absoluto (6) o endereçamento é definido pelo byte seguinte ao código da instrução mais três bits localizados no próprio código. Da mesma forma de endereçamento anterior, esta depende da posição atual para calcular o endereço desejado. Os 11 bits resultantes da instrução substituem os 11 bits menos significativos do registrador PC gerando o novo endereço. Deve-se observar que este endereçamento fica limitado a 2kbytes de distância da instrução atual.

Exemplo de instrução; ACALL 0x2100

No modo Longo (7) o endereço é inteiramente definido nos dois bytes seguintes a instrução. Neste caso o próximo byte imediatamente após o código da instrução contém a parte alta do endereçamento e o byte seguinte a parte baixa (menos significativa).

Exemplo de instrução: LJMP 0x2000

No modo Indexado (8) o endereçamento é feito pelo valor armazenado em um registrador mais o conteúdo do registrador . Este forma visa auxiliar na leitura de tabelas em memórias, pois o registrador principal contém o endereço inicial da tabela e o registrador A contém o índice a ser acessado.

Exemplo de instrução: MOVC A, @A+DPTR

2.7.2 Tipos de instruções

Segundo Nicolosi (2002) pode-se classificar as instruções em cinco grupos para melhor compreensão. Estes cinco grupos são:

1. Instruções Aritméticas
2. Instruções Lógicas
3. Instruções de Transferência de Dados
4. Instruções *Booleanas*

5. Instruções de Desvio

O anexo A contém a definição de cada instrução com seu tamanho em bytes, ciclos de máquina para execução e *flags* afetados.

3 SDCC – SMALL DEVICE C COMPILER

3.1 O que é o SDCC

O SDCC é um compilador direcionado para microcontroladores com arquiteturas de 8 bits. Trata-se de um projeto de código fonte aberto que possui suporte para as plataformas MCS51, Zilog Z80, Dallas DS80C390 e algumas versões do Microchip PIC (NICOLOSI e BRONZERI, 2005). Teve início em 1999 e atualmente a versão mais recente lançada como estável é a 3.2.0 .

Uma coleção de ferramentas é desenvolvida junto ao SDCC como emulador, conversor, *debugger*, etc. (SDCC, 2012).

3.2 Tipos de dados suportados pelo SDCC

Os tipos de dados simples suportados pelo SDCC são descritos na Tabela 4.

Tabela 4 - Tipos de dados suportados pelo SDCC

Tipo	Número de Bits	Número de Bytes	Limite de valores
char	8	1	-128 a +127
short	16	2	-32768 a +32767
int	16	2	-32768 a +32767
long	32	4	-2147483648 a +2147483647
float	32	4	3,4E-38 a3,4E+38

Fonte: Adaptado de (NICOLOSI; BRONZERI, 2005)

Da mesma forma que o ANSI C, existem modificadores para os tipos de dados permitindo suprimir a faixa negativa de valores. A Tabela 5 apresenta estes modificadores com seus tamanhos e faixas(NICOLOSI e BRONZERI, 2005).

Tabela 5 - Modificadores de tipos de dados

Tipo	Bits	Bytes	Limite de valores
unsigned char	8	1	-128 a +127
unsigned short e unsigned int	16	2	-32768 a +32767
Unsigned long	32	4	-2147483648 a +2147483647

Fonte: Adaptado de (NICOLOSI; BRONZERI, 2005)

A criação de programas se dá de forma similar ao ANSI C padrão. Na Listagem 1 tem-se um exemplo de programa escrito com chamadas de funções e declarações de variáveis.

Listagem 1 - Exemplo de programa escrito em C

```
1       #include <at89x52.h>
2       int i;
3       int foo(void) {
4         i++;
5         return i*3;
6       }
7       void main(void) {
8         i = 5;
9         P1 = foo();
10        while(1);
11      }
```

Fonte: Elaborado pelo autor.

Nota-se que este programa não tem utilidade prática, mas pode-se fazer uma análise didática interessante. Na linha 1 observa-se a inclusão de uma biblioteca que possui declarações e funções específicas para o microcontrolador da Atmel AT89S52. O compilador SDCC possui várias bibliotecas para diferentes versões dos chips para o qual possui suporte. Na linha 2 é declarada a variável i do tipo int, que será utilizada nas linhas posteriores. Na linha 3 existe a declaração de uma função denominada *foo* que retorna um tipo de dado *int* e não recebe parâmetros. Esta função incrementa a variável global i e retorna o seu valor multiplicado por três. Na linha 7 observa-se a função principal do qual todo programa deve conter. Trata-se da função *main* que é onde o programa tem o início de sua execução. No exemplo, linha 8, a função *main* está atribuindo o valor cinco à variável *i.* Na linha 9 a função *foo* é chamada e seu resultado atribuído à porta P1 do microcontrolador. A linha 10 faz com que o programa pare a sua execução pois chegou ao final.

Porém programas escritos para executar em sistemas embarcados, como é caso, devem sempre ficar presos em laços de repetição realizando alguma tarefa até que sejam desligados ou reiniciados, pois diferentemente de um programa tradicional para computador, estes não possuem um sistema operacional para o qual podem retornar após o término da execução.

3.3 Alocações de variáveis

Conforme citado no capítulo 2, o 8051 possui três regiões de memória distintas, sendo estas a memória de código, a memória interna de dados (RAM) e a memória externa de dados (RAM). Surge a partir desta característica a necessidade de informar em qual destas memórias a variável a ser criada deve ter os seus dados alocados. Para selecionar a região de memória no qual uma variável a ser declarada deve ser alocada, o SDCC possui palavras chaves a serem descritas antecedendo o tipo de dado da variável (NICOLOSI; BRONZERI, 2005).

Para uma variável ser alocada na memória interna de acesso direto utiliza-se a palavra-chave *data*. Quando uma declaração de variável não definir explicitamente sua região de alocação, esta será definida pelo compilador.

Declarar uma variável cujo conteúdo deve estar na região de acesso indireto da memória (8052) utiliza-se a palavra-chave *idata*.

Para uma variável ser alocada em memória RAM externa, a palavra-chave na declaração é *xdata*.

Existe ainda a opção de variáveis em memória de código, sendo que estas serão de apenas leitura, não havendo a possibilidade de modificação de seu conteúdo em tempo de execução. Na declaração deste tipo de variável utiliza-se a palavra-chave *code*.

3.4 Arquivo de depuração CDB

O SDCC gera um arquivo com informações de depuração quando a instrução *–debug* é informada em tempo de compilação. Este arquivo contém toda a informação que descreve variáveis, funções, linhas e itens de memória. (STORY, 2003).

Estas informações permitem utilitários externos localizar e interpretar funções, variáveis e tipos de dados. Ferramentas de desenvolvimento utilizam estes dados para analisar e descrever o código fonte de alto nível em relação à execução do código gerado pelo compilador.

Para cada arquivo fonte gera-se o respectivo arquivo CDB, sendo que o principal objeto que gera o código executável final contém todas as informações referente ao programa compilado.

As otimizações automáticas impostas pelo compilador devem ser desabilitadas a fim de impedir o mascaramento de variáveis ou endereçamentos que podem dificultar a depuração do código.

Existem 11 tipos de registros definidos para o arquivo de depuração (SDCC, 2012). Cada tipo de registro descreve uma estrutura distinta do vínculo entre o fonte do programa e o código executável gerado.

A Tabela 6 demonstra quais os registros existentes.

Tabela 6 - Tipos de registros do arquivo CDB

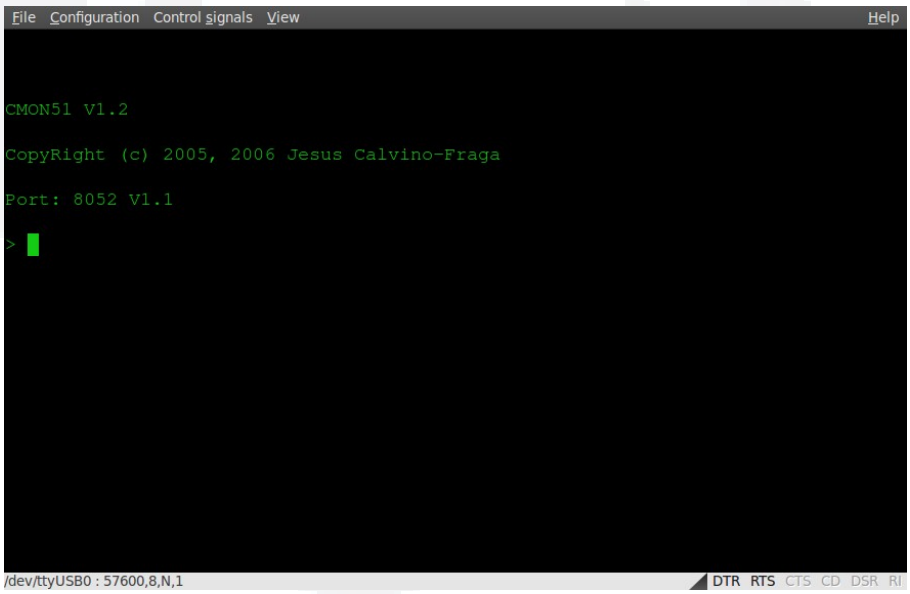
Módulo	ID	Descrição
Module	M	Descreve a nomenclatura do arquivo fonte de referência
Symbol	S	Todos os símbolos com nomes, sendo variáveis locais, globais e parâmetros.
Type Chain	-	Define o tipo de dado presente em um símbolo
Function	F	Indica uma função presente no arquivo fonte
Type Record	T	Define tipos de variáveis complexos como *unions* e *structs*
Type Member	-	Auxilia na definição dos tipos citados no item anterior
Link Address of Symbol	L	Define endereço de um símbolo
Linker ASM Line Record	L:A	Define vínculo entre linhas do arquivo assembly e o endereço das instruções
Linker C-Line Record	L:C	Define vínculo entre linhas do arquivo fonte C e o endereço das instruções

Fonte: Adaptado de (CDB, 2003).

4 CMON51

O CMON51 é um programa monitor (depurador) para a família 8051 escrito por Jesus Calvino-Fraga e que tem seu código fonte aberto disponível em seu site oficial(FRAGA, 2005). Sua interface se apresenta em modo texto e para interação com o aplicativo deve ser utilizado um programa de emulação de terminal . A Figura 10 mostra a mensagem inicial do programa em um terminal *GtkTerm* (FRAGA, 2005).

Figura 10 - Tela de inicial do CMON51



Fonte: Elaborado pelo autor.

O CMON51 permite as seguintes funcionalidades sobre o microcontrolador:

- Examinar e modificar a memória RAM interna e externa;
- Examinar a memória de código;
- Examinar e modificar qualquer registrador incluindo os SFRs;
- Baixar um programa em formato Intel Hex para a memória;
- Visualizar o código assembly na memória;
- Executar o programa e operações de depuração como passo-a-passo;
- Usar breakpoints;

Por se tratar de um software escrito para executar sobre o próprio microcontrolador, existem alguns recursos que ficam reservados para o CMON51 funcionar:

- A porta serial com o T/C 1 como gerador de taxa;
- O vetor de interrupção do Timer 1;
- 256 bytes de memória externa para variáveis;
- 8 Kbytes de memória de código;

4.1 Visualizando e editando a memória

O CMON51 disponibiliza comandos para visualizar a memória de código e as memórias RAM internas e externas. Para tal utiliza-se o comando X [endereço em hexadecimal] [tamanho segmento] para visualizar a memória externa, o comando D para visualizar a memória interna de acesso direto, o comando I para visualizar a a memória interna de acesso indireto e o comando C [endereço em hexadecimal] [tamanho segmento] para visualizar o conteúdo da memória de código (ROM).

A modificação dos valores de em memórias RAM pode ser alterado pelos comandos MX [endereço] para modificar a memória RAM externa, MD [endereço] para modificar a memória de acesso direto e MI [endereço] para modificar a memória de acesso indireto.

4.2 Inserindo *breakpoints* no código

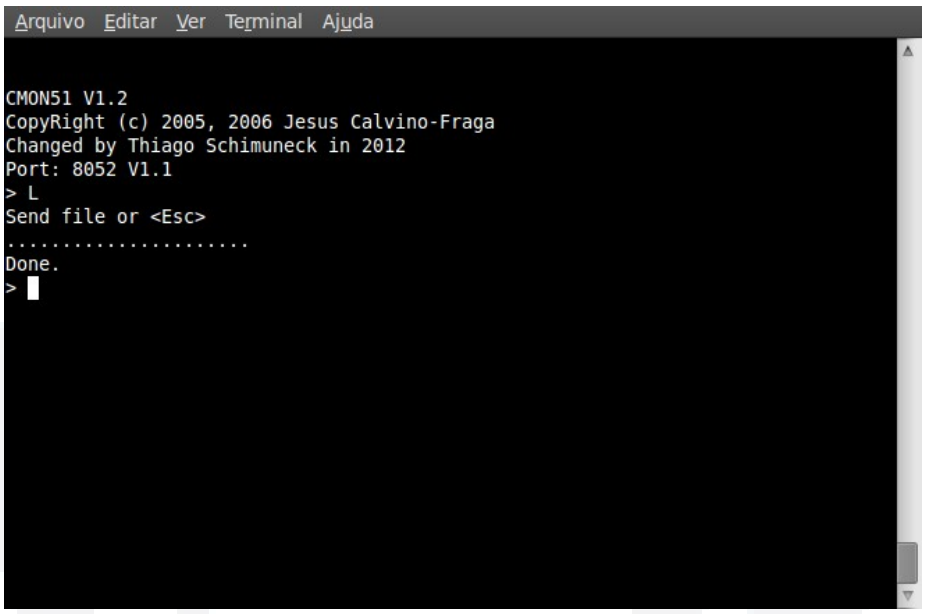
Ao iniciar o programa a ser depurado através dos recursos do CMON51, este segue sua execução normal até o momento em que o usuário solicita a parada ou ao encontrar uma instrução de *breakpoint* inserida no código. Para tal efeito é necessário a inserção da instrução LCALL 001Bh exatamente uma instrução acima do ponto em que se deseja parar(FRAGA, 2005).

Existe ainda a opção de *breakpoint* por software onde até quatro endereços podem ser definidos para que o CMON51 interrompa a execução no momento em que o fluxo de execução coincida com este endereço. Este modo de parada do programa via software não é utilizada pela aplicação desenvolvida neste trabalho.

4.3 Envio do código executável

O envio do código executável para o hardware é realizado pelo CMON51 através do comando *L*. A imagem do programa deve ser enviada no formato Intel Hex sendo a transferência em ASCII puro. A Figura 11 demonstra a tela após transferência imagem de código para o hardware.

Figura 11 - Tela de inicial do CMON51



Fonte: Elaborado pelo autor.

5 SOLUÇÕES EXISTENTES

Dentre as buscas por soluções já existentes, duas merecem destaque, sendo a primeira o software uVision desenvolvido pela empresa Keil, e o MCU 8051 criado por Martin Osmera (LUCAS, 2012).

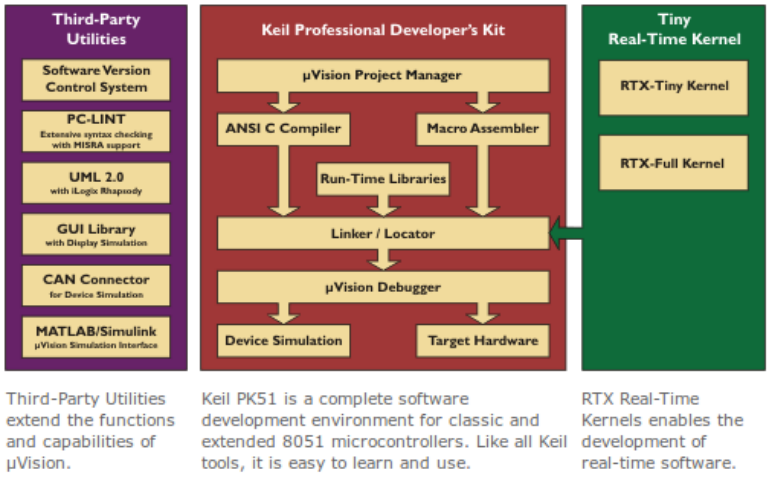
5.1 Keil uVision

O uVision possui suporte a diversas variantes do 8051, dentre eles podemos citar Acer Labs M6032, Actel Core8051, Analog Devices ADuC812, Atmel AT89C51, Cypress EZ-USB FX2 (CY7C68XXX), Dallas Semiconductor DS89C420, Infineon XC888CLM-8FF, Intel 87C51, NXP P89V51RD2, Silicon Laboratories, Inc. C8051F360.

A versão 9.05 e possui integrado compilador C, compilador assembly, ligador, depurador, simulador e conexão para placas de desenvolvimento alvo, permitindo emulação em tempo real (KEIL, 2012).

A Figura 12 apresenta o conjunto de utilitários disponível na ferramenta.

Figura 12 - Arquitetura do Keil uVision



Fonte: Adaptado de (KEIL, 2012).

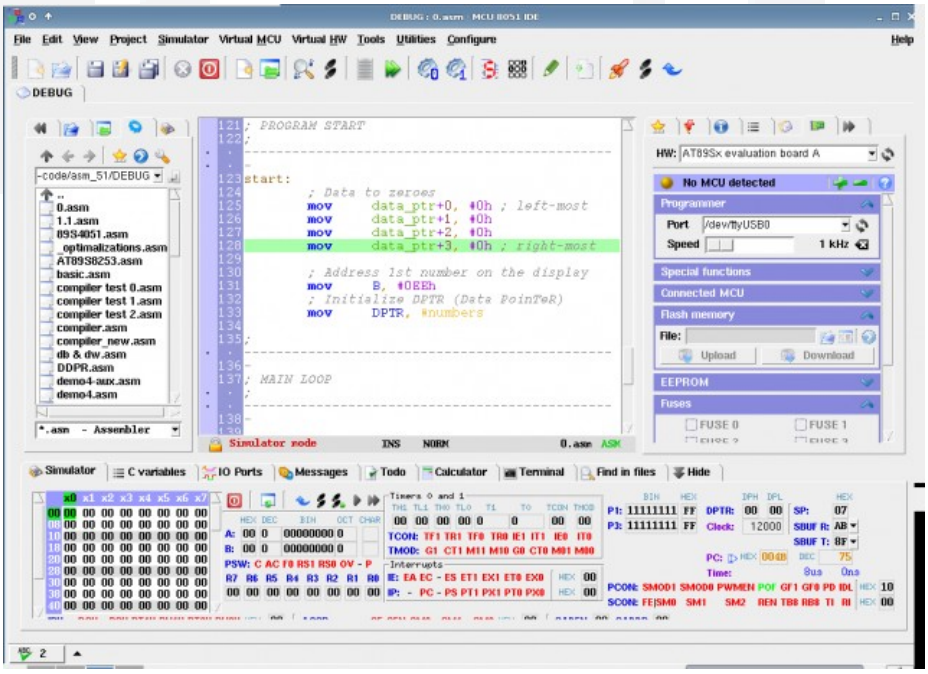
Esta ferramenta é proprietária, disponibilizando para download uma versão de demonstração onde é possível criar programas com até 2 kilobytes de tamanho. Existe

hardware específico que permite a emulação do software desenvolvido em tempo real (KEIL, 2012).

5.2 MCU 8051 IDE

O MCU 8051 trata-se de uma IDE desenvolvida exclusivamente para suporte ao tradicional 8051. Criado e atualmente mantido por Martin Osmera, o projeto é de código aberto e está escrito na linguagem tcl/tk. Possui suporte a mais de 79 microcontroladores compatíveis com a família MCS51, um avançado simulador do 8051, editor de código avançado com avançado sistema de destaque de sintaxe e auto completador, macro assembler incorporado, suporte ao SDCC, simulação de hardwares como displays de 7 segmentos e teclados, suporte a gravadores ISP, etc (LUCAS, 2012). A Figura 13 mostra a interface da IDE.

Figura 13 - Ambiente do MCU 8051 IDE



Fonte: (MCU 8051 IDE, 2012).

Esta ferramenta não possui limitações na versão de download por ser uma aplicação de código fonte aberto, porém esta não possui suporte à emulação de código em tempo real no hardware.

6 SOLUÇÃO DESENVOLVIDA

A solução desenvolvida é uma interface de ambiente de desenvolvimento onde o usuário pode criar projetos de forma transparente e com apoio e facilidades que uma ferramenta de desenvolvimento tem a oferecer. O software CMON51 será utilizado para gerenciar o código executável de programas escritos para o 8051 trazendo possibilidade de inserir *breakpoints*, iniciar, parar e executar passo-a-passo o programa em tempo real. O compilador SDCC fará a compilação do código escrito em linguagem C para a família MCS51 traduzindo este para o código executável carregado e gerenciado pelo CMON51.

Devido o fato da arquitetura da família MCS51 não possuir pinos exclusivos para o envio e depuração de código diretamente no chip, utiliza-se alguns recursos genéricos do microcontrolador para permitir esta emulação (CMON51, 2005).

Tendo o CMON51 como base para suportar a emulação do software, o usuário deverá respeitar certas regras ao escrever seus programas e não poderá utilizar alguns recursos, sendo eles a UART interna e o temporizador 1 do chip, incluindo os seus respectivos vetores de interrupção. O código executável gerado pelo usuário deverá estar alocado a partir do endereço 2000h para ter seu funcionamento garantido e a porção de 256 bytes de memória externa localizada no endereço 7F00h fica reservada para o programa monitor e não pode ser alterada pelo usuário. Apenas estarão disponíveis os pinos da porta P1 e alguns pinos da porta P3 para utilização de propósito geral da aplicação.

Esta estrutura proposta se torna compatível com os kits de desenvolvimento hoje utilizados pela UNIVATES. Tais kits possuem disponíveis vários periféricos como teclado numérico, display alfanumérico, conversor analógico-digital e conversor digital-analógico. todos mapeados em memória externa de forma a não interferir no funcionamento do básico do monitor CMON51.

Alterou-se então o programa monitor atual presente nestes kits pelo código do CMON51 de modo permitir a utilização do novo ambiente ocorrer de forma transparente sem qualquer alteração no projeto esquemático do kit.

6.1 Alterações no software CMON51

Para permitir uma iteração mais estável com o CMON51, duas alterações foram necessárias em seu código original, uma incluindo novo comando para reinicializar via software o equipamento, sendo chamado de “RST”, e outra reprogramando a interrupção do canal serial do microcontrolador a fim de possibilitar a parada do programa em execução quando solicitada pelo usuário. Para reinicialização foram utilizadas as definições descritas pela documentação do microcontrolador, que descrevem quais registradores são modificados por uma reinicialização via hardware. A Listagem 2 exibe o trecho de código inserido no arquivo cmon51.c.

Listagem 2 - Reinicialização do hardware

```
1. case ID_rst:
2.     fillmem((unsigned char data *)br, 8, 0);
3.     __asm
4.         clr a
5.         mov ie, a
6.         mov tcon, a
7.         mov t2con, a
8.         mov scon, a
9.         mov b, a
10.             mov sp, a
11.             mov psw, a
12.             mov th0, a
13.             mov tl0, a
14.             mov th1, a
15.             mov tl1, a
16.             mov tmod, a
17.             mov rcap2h, a
18.             mov rcap2l, a
19.             mov tl2, a
20.             mov th2, a
21.             mov ip, a
22.             mov dptr,#0x0000
23.             cpl a
24.             mov p0, a
25.             mov p1, a
26.             mov p2, a
27.             mov p3, a
28.             anl ip, #0x0C
```

```
29.             anl pcon, #0x70
30.             clr a
31.             jmp @a+dptr
32.         __endasm;
```

Fonte: Elaborado pelo autor.

A reprogramação dos vetores de interrupção interfere nos endereços 0000h e 001Bh e 0023h de forma a permitir a parada do programa em execução assim que uma atividade seja percebida na porta serial do microcontrolador. A Listagem 3 demostra o código final do trecho de código no arquivo 8052.c.

Listagem 3 - Vetores de interrupção do CMON51

```
1. _asm
2. .area JUMPCSEG   (ABS,CODE)
3.
4. .org 0x0000
5.     ljmp __sdcc_gsinit_startup
6.
7. .org 0x0003
8.     ljmp XRAM_CODE_LOC+0x03
9.
10.      .org 0x000B
11.          ljmp XRAM_CODE_LOC+0x0B
12.
13.      .org 0x0013
14.          ljmp XRAM_CODE_LOC+0x13
15.
16.      ;Vector 0x001B used by cmon51!
17.      .org 0x001B
18.          ljmp _step_and_break
19.
20.      .org 0x0023
21.          jnb RI, _ignore_tx
22.          ljmp _step_and_break
23.          ;ljmp XRAM_CODE_LOC+0x23
24.
25.      .org 0x002B
26.          ljmp XRAM_CODE_LOC+0x2B
27.
28.      _ignore_tx:
29.         clr TI
30.         reti
```

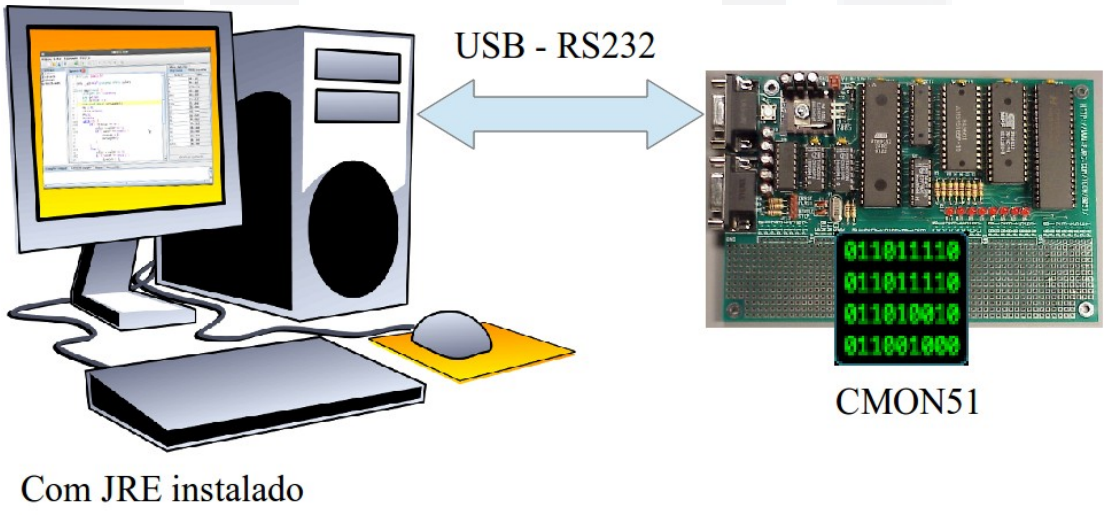
```
31.
32.       _endasm;
```

Fonte: Elaborado pelo autor.

6.2 Desenvolvimento do ambiente

De posse do conhecimento da ferramenta SDCC para a compilação dos programas fontes, tendo como alvo a plataforma 8051, e da ferramenta CMON51, para a gerência dos programas em tempo de execução, concentrou-se os esforços no desenvolvimento de uma interface gráfica, conforme proposta inicial, sob uma plataforma de desenvolvimento que permite o intercâmbio entre sistemas operacionais mais populares. Neste caso optou-se pelo Java devido à familiaridade do ambiente e a sua estabilidade. A Figura 14 apresenta um esboço da estrutura necessária. (JAVA, 2013b).

Figura 14 - Componentes do projeto



Fonte: Elaborado pelo autor.

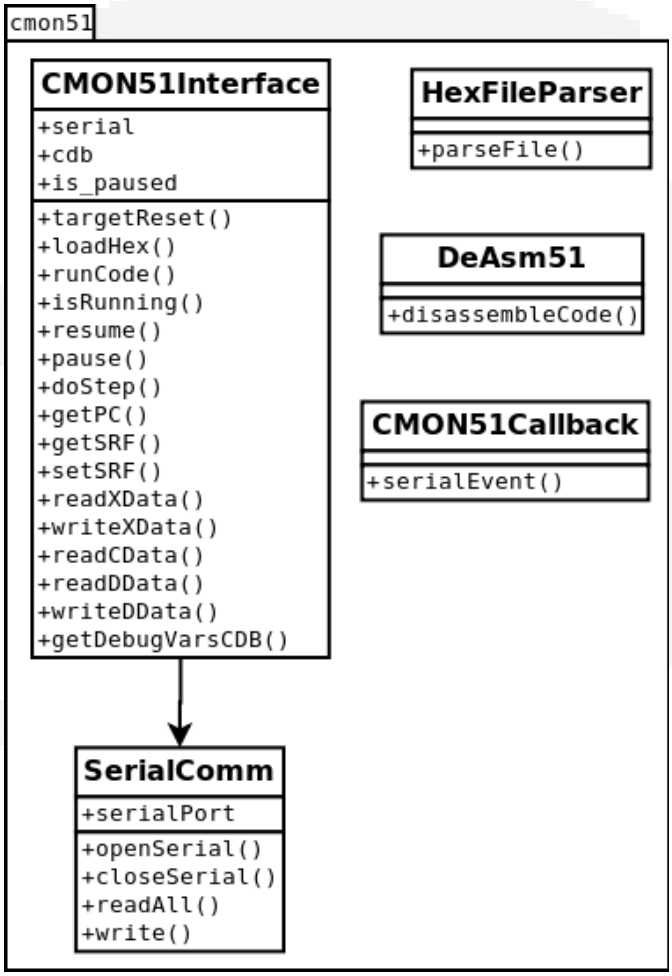
O passo inicial para a construção foi a escrita de classes que permitem fazer a comunicação serial com o hardware do kit de desenvolvimento e o ambiente computacional. Para tal, a disponibilidade de comunicação serial do ambiente Java é necessária. Porém, este ambiente possui suporte nativo apenas os sistemas operacionais Solaris SPARC, Solaris x86, e Linux x86 o que torna necessário a utilização da biblioteca RXTX (JARVI, 2013a).

Sobre este conjunto inicial de classes implementou-se uma série de funções que permitem abstrair os comandos necessários para manipular o CMON51. Desta forma é

possível solicitar o envio do programa para o kit através de um comando único, por exemplo, ou ler e modificar variáveis ou registradores do microcontrolador de uma forma simplificada, facilitando a integração com a interface gráfica.

A velocidade de comunicação da porta serial é fixada nas classes desenvolvidas em 57600 bps. Já o kit de desenvolvimento utilizado pela UNIVATES trabalha na frequência de 11,0592 MHz, o que configura uma velocidade máxima de comunicação de 57,6 kbps. Não é possível a alteração desta taxa pelo usuário. A Figura 15 exibe as principais classes e métodos utilizadas para esta comunicação.

Figura 15 - Classes do pacote CMON51

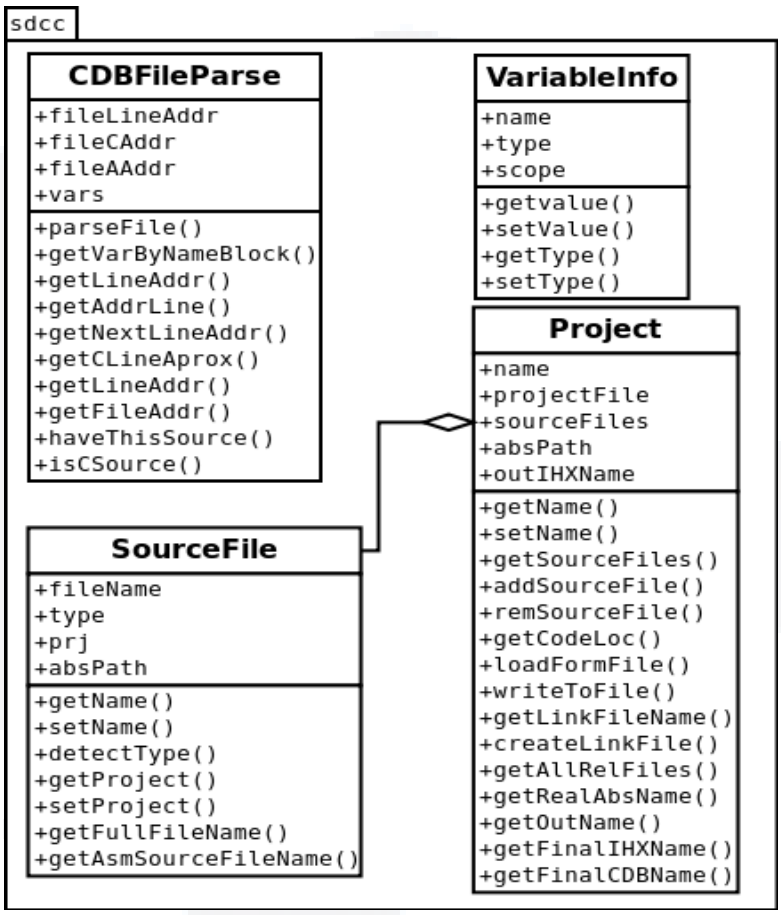


Fonte: Elaborado pelo autor.

A análise do programa compilado e em execução pode ser realizada fazendo a análise do arquivo CDB gerado pelo compilador. Logo, um conjunto de classes foi desenvolvido a fim de analisar e retirar as informações necessárias para que a interface possa informar ao usuário informações sobre variáveis, registradores e linha atual de execução de forma a permitir funções básicas de depuração. A estrutura do arquivo CDB já foi brevemente

comentada na Subseção 3.4, porém para a necessidade prevista para o projeto utiliza apenas parte das informações disponíveis em particular, a posição das instruções no arquivo do programa final em relação às linhas no arquivo fonte e os endereços das variáveis e registradores alocadas na memória. A Figura 16 exibe as principais classes e métodos para interação com o arquivo CDB e o compilador.

Figura 16 - Classes de interação com o arquivo CDB e compilador

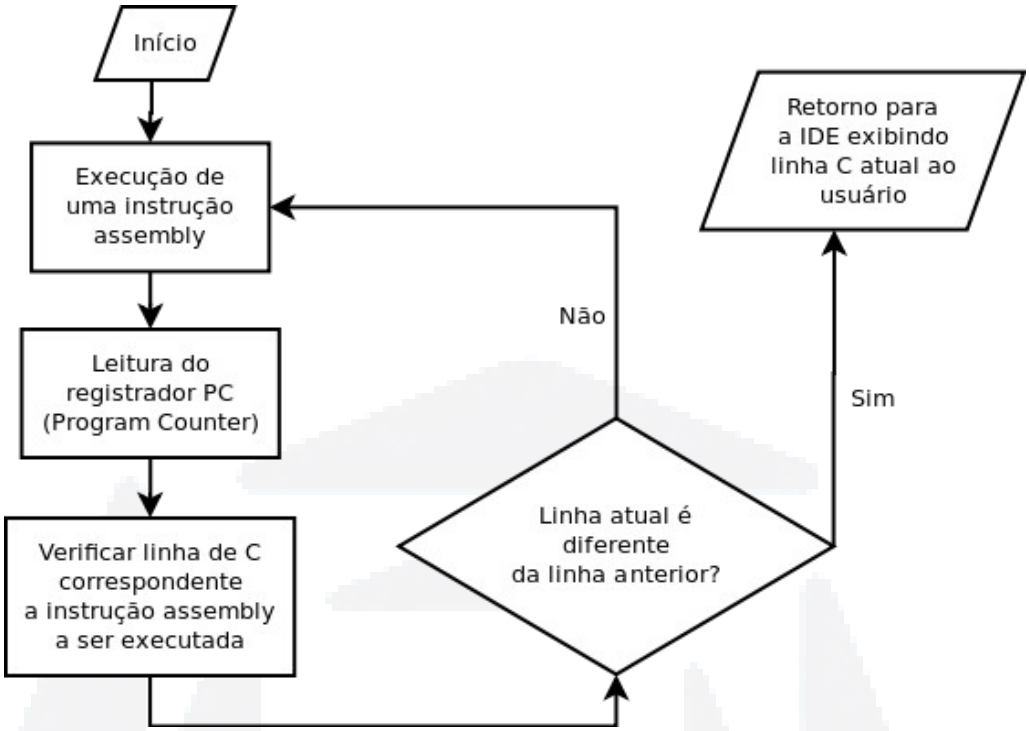


Fonte: Elaborado pelo autor.

Tanto o código para interação com o CMON51 através da porta serial quanto a análise do arquivo de depuração foram escritos para serem destacáveis do código da interface, permitindo que possam ser utilizados no desenvolvimento de ambientes em outras plataformas como, por exemplo, *plugins*.

A operação passo-a-passo do código em C foi implementada utilizando um algoritmo especial que pode ser observado na figura 17.

Figura 17 - Janela do ambiente

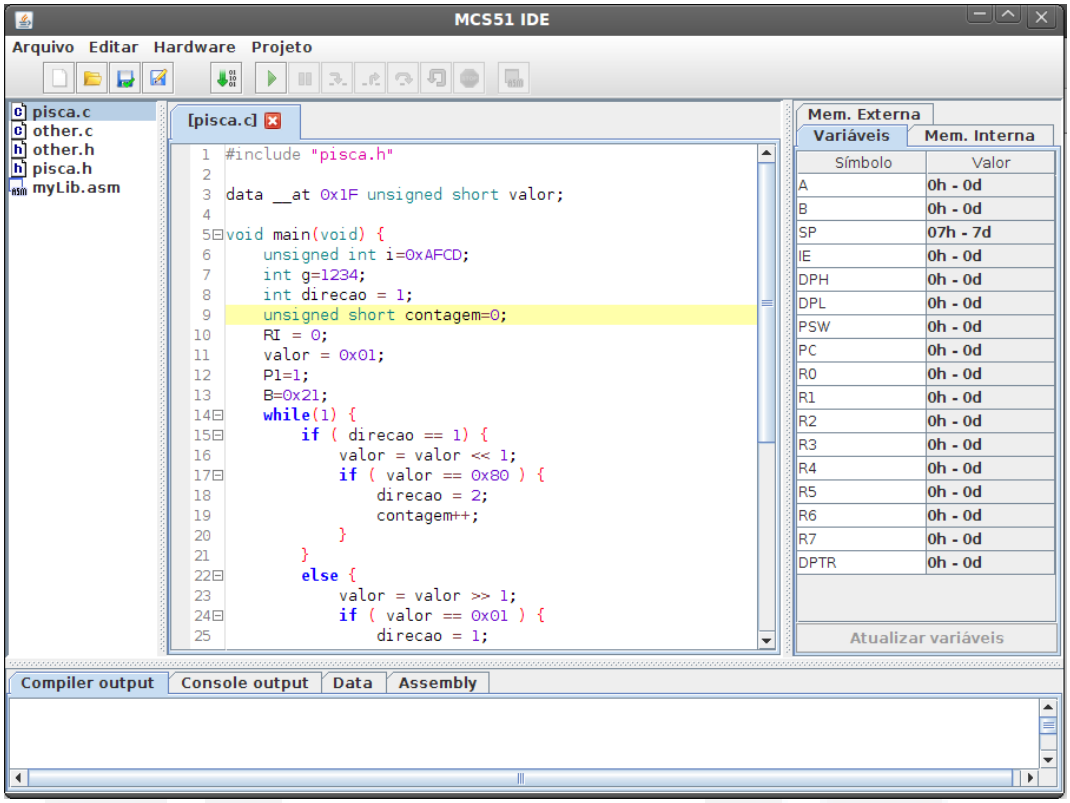


Fonte: Elaborado pelo autor.

6.3 A interface gráfica e sua operação

A interface do ambiente de desenvolvimento foi totalmente desenvolvida usando a biblioteca gráfica nativa do Java. Nesta constam uma barra de menus onde estão disponíveis comandos para operar com arquivos e configurar o ambiente. Logo abaixo uma barra de ferramentas com botões de atalho para manipulação de arquivos, compilação do código, execução e depuração deste no hardware está disponível. Na esquerda uma janela contém os arquivos do projeto atualmente aberto permitindo navegar entre estes. A parte central concentra o editor de texto com destacador de texto e numeração de linhas. Ao clicar na barra anterior ao número da linha podemos marcar um *breakpoint* no código fonte do programa. Também é neste local que é indicado a linha atual de execução através de uma seta. À direita existem três abas que permite a visualização e ajuste de valores em memória ou em variáveis. E finalmente na parte inferior da janela do ambiente tem-se a saída do compilador, permitindo a localização de erros no fonte durante o processo de compilação ou ligação. A Figura 18 mostra a janela do ambiente.

Figura 18 - Janela do ambiente



Fonte: Elaborado pelo autor.

O ambiente permite que sejam adicionados vários arquivos-fontes em um projeto e que sejam compilados e ligados juntos gerando, apenas um executável. Para tal o usuário deve abrir o código fonte no ambiente e utilizar a opção *Adicionar arquivo ao projeto* localizada no menu *Projeto*. Na sequência, salva-se o projeto utilizando a opção *Salvar* no mesmo diretório onde estão localizados os arquivos fontes adicionados. As únicas restrições para a compilação em vários arquivos fontes é que todos devem estar dentro do mesmo diretório e terem nomes distintos independentemente de caracteres maiúsculos e minúsculos. Os códigos fontes abertos no editor possuem os caracteres [e] em torno do seu nome na aba de editores.

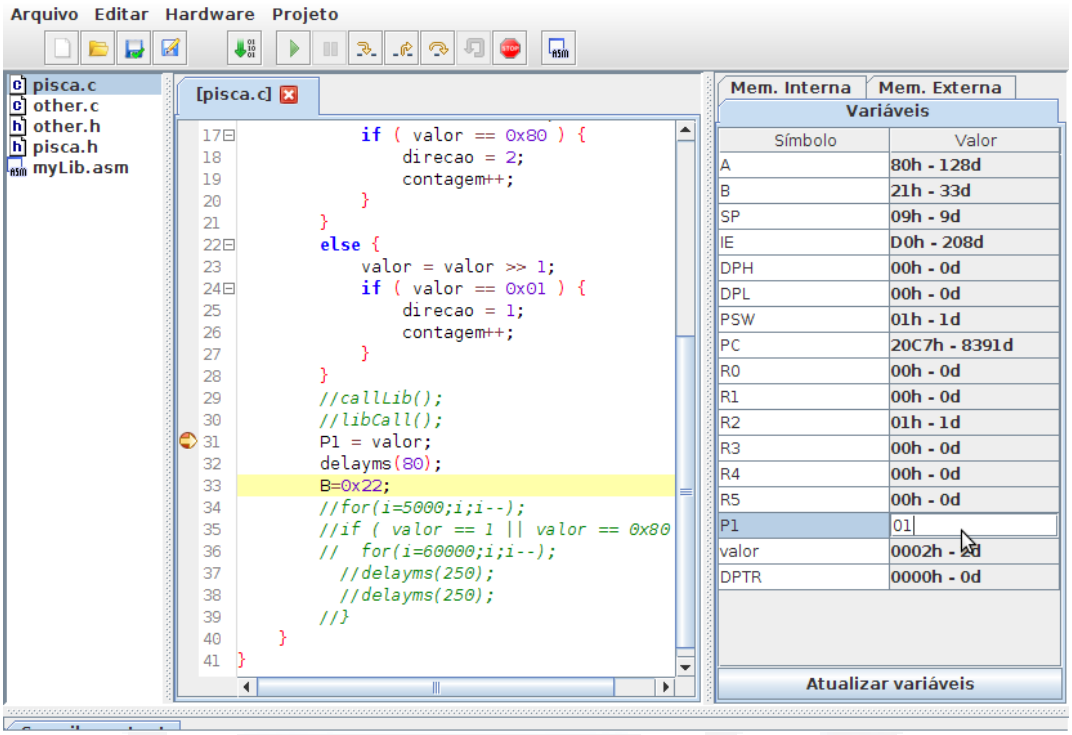
Para códigos fontes singulares a compilação e depuração ocorre de forma direta e transparente.

O controle da compilação, execução, interrupção e parada do código no hardware ocorre pelos botões localizados na barra superior, sendo estes habilitados de acordo com o estado do programa em execução no kit.

A inserção de *breakpoints* ocorre ao se clicar na região à esquerda da numeração da linha. Um símbolo sobre a linha indica que a execução irá parar ao atingir esta parte do código. Um número indeterminado de *breakpoints* pode ser inserido no código, lembrando

que cada um gera um incremento de três bytes no programa final que será enviado ao kit de desenvolvimento. A Figura 19 demonstra a execução de um programa exemplo aguardando interação do usuário em um *breakpoint*. Nota-se que na direta os registradores principais e variáveis podem ter seus valores examinados e alterados, assim como a memória interna e externa.

Figura 19 - Componentes do projeto

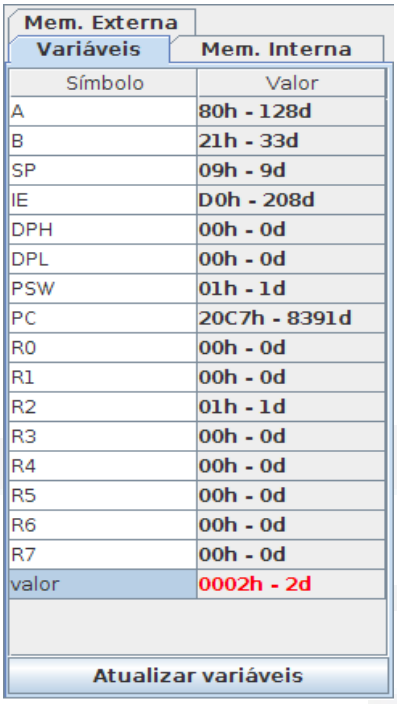


Fonte: Elaborado pelo autor.

Na barra de botões superior existe o botão *ASM* o qual permite a visualização do código em assembly gerado pelo compilador. Esta função permite visualizar qual a instrução de baixo nível será executada e qual a implementação feita pelo compilador para a rotina em questão.

Para a manipulação do espaço de memória, variáveis e registradores, utiliza-se a interface localizada à direita do ambiente. A aba de variáveis permite a visualização de registradores e variáveis declaradas em C, sendo seu conteúdo apresentado em base hexadecimal e decimal. Pode-se ajustar o conteúdo das variáveis com um clique duplo sobre o valor atual desta, lembrando que a inserção pelo usuário deve sempre ser feita em base hexadecimal. A Figura 20 exibe o formato da tela.

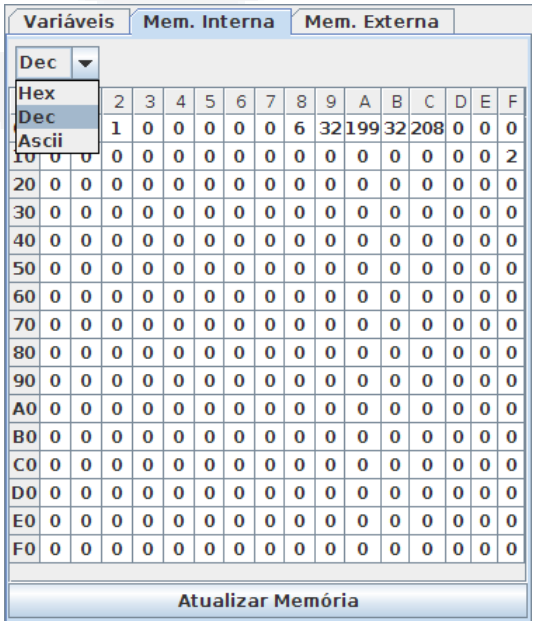
Figura 20 - Janela para valores de variáveis



Fonte: Elaborado pelo autor.

Na aba de *Memória Interna* tem-se a possibilidade de visualização e edição dos valores dos primeiros 128 bytes da memória acessados direta ou indiretamente e os 128 bytes superiores acessados indiretamente conforme descrito na seção 2.2. A Figura 21 exibe a aparência desta janela onde nota-se que é possível alterar a base de exibição dos valores.

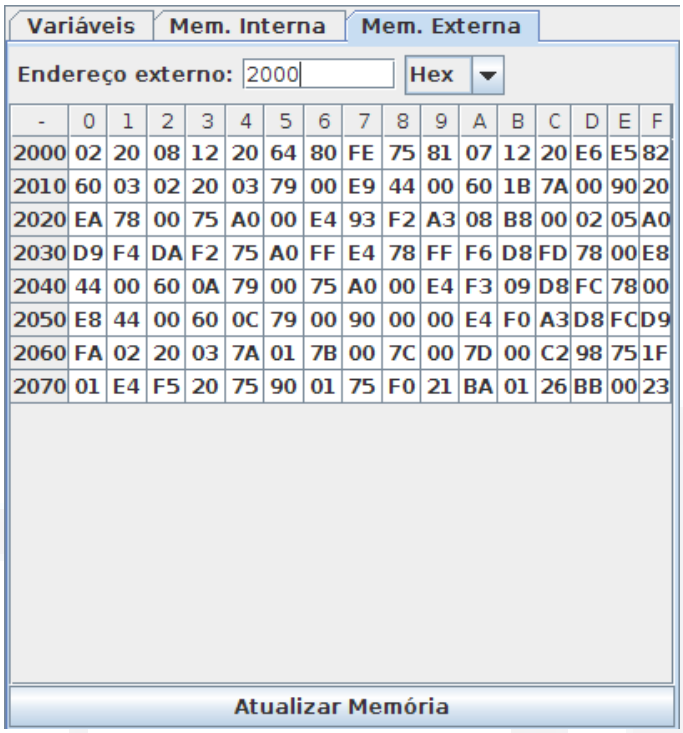
Figura 21 - Janela para memória interna



Fonte: Elaborado pelo autor.

Tem-se ainda a visualização e edição da memória externa. Como neste caso existem 64Kbytes de memória disponíveis, sendo que apenas uma porção desta é exibida, cujo endereço inicial é definido pelo campo na parte superior da aba. A Figura 22 exibe a aparência desta interface.

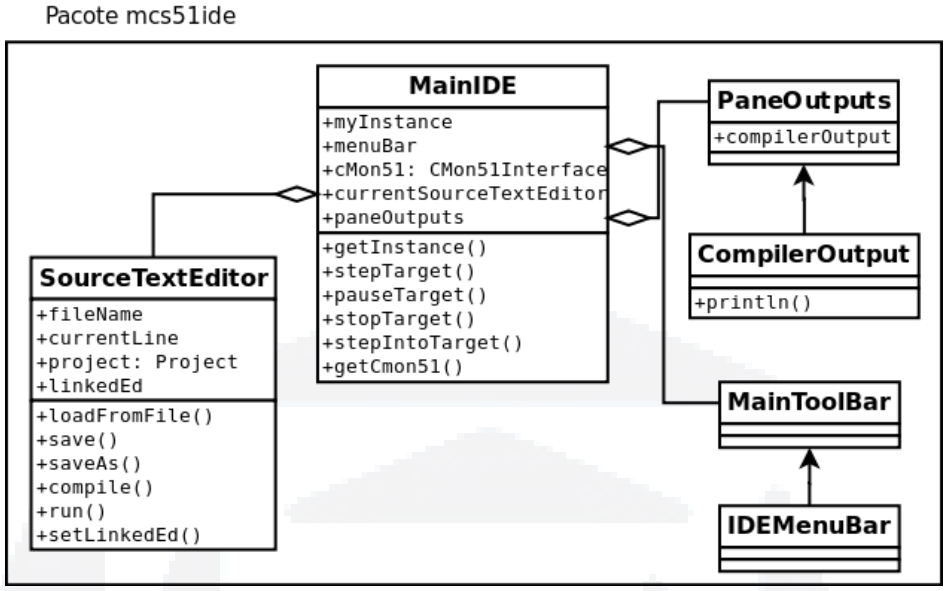
Figura 22 - Janela para memória interna



Fonte: Elaborado pelo autor.

As principais classes e métodos que compõem a interface gráfica podem ser visualizadas na Figura 23.

Figura 23 - Classes da interface gráfica



Fonte: Elaborado pelo autor.

7 EXPERIMENTO

Com o objetivo de validar a solução apresentada, utilizou-se um dos kits da UNIVATES, substituindo o microcontrolador com o antigo software monitor pelo utilizado na placa experimental contendo o CMON51.

7.1 Descrição do hardware do kit

O kit de desenvolvimento (KID51) utilizado no experimento prático inclui um chip AT89S52 da Atmel (*core* 8052), com uma memória RAM estática de 32Kbytes sendo esta espelhada entre a memória de dados externa e a memória de código externa a partir do endereço 2000h. O cristal para gerar a frequência de operação (57600 bps) do microcontrolador neste kit é de 11,0592 MHz.

O KID51 possui display LCD alfanumérico de 2 linhas por 16 colunas, teclado numérico de 12 teclas, conversor analógico-digital de 12 bits (4 canais) e conversor digital-analógico de 10 bits (2 canais).

7.2 Validação sobre o kit de desenvolvimento da UNIVATES.

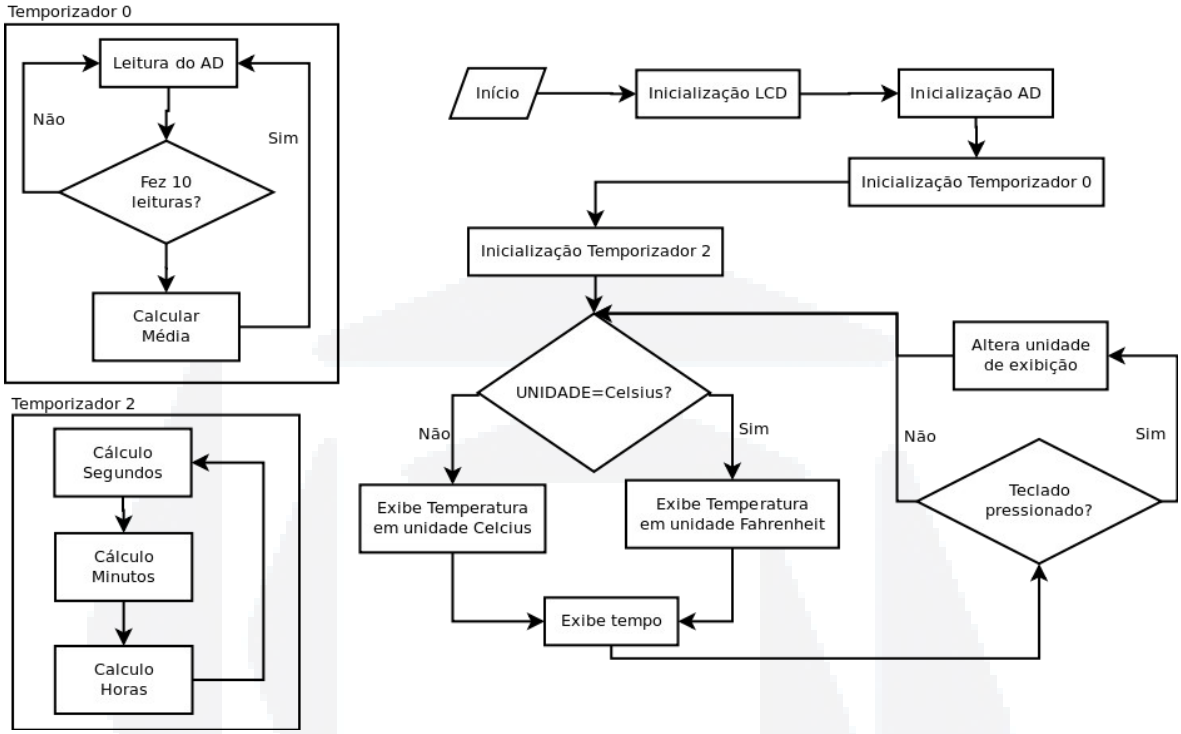
Para a validação da ferramenta proposta foi desenvolvido um programa em C que faz acesso aos principais periféricos deste kit. Basicamente a função do programa é realizar a leitura do canal analógico-digital disponível, onde foi conectado um sensor de temperatura. No programa de validação foram realizadas 10 aquisições por segundo, calculado a temperatura média entre as aquisições, feita a conversão para o valor de temperatura de acordo com a unidade selecionada e exibido este valor no display LCD.

Para tal efeito implementou-se uma biblioteca para interação com o display LCD permitindo sua inicialização, configuração e escrita neste.

O tempo decorrido de leitura do conversor analógico é contabilizado utilizando-se um temporizador de 16 bits, com cálculo de hora, minuto e segundo e apresentado do display LCD. A leitura do teclado é feita para que o usuário possa selecionar a unidade da temperatura apresentada (exibição em graus Celsius ou Fahrenheit). A figura 24 exibe o

diagrama com os fluxos de execução do programa. Nota-se que interrupções são utilizadas para a paralelização das etapas envolvidas.

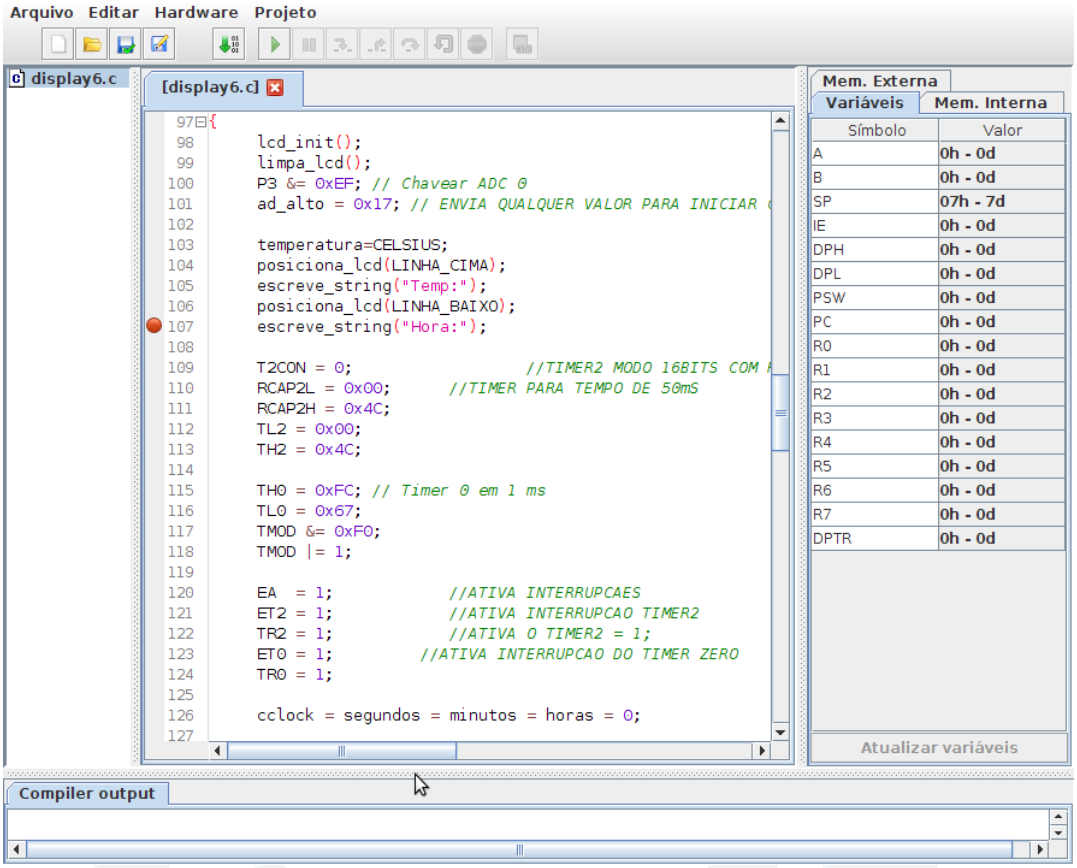
Figura 24 - Diagrama do programa para validação



Fonte: Elaborado pelo autor.

Na Figura 25 é apresentada a interface desenvolvida contendo o programa de validação. Na aba da direita se observa os registradores enquanto que o código principal fica localizado no centro da tela.

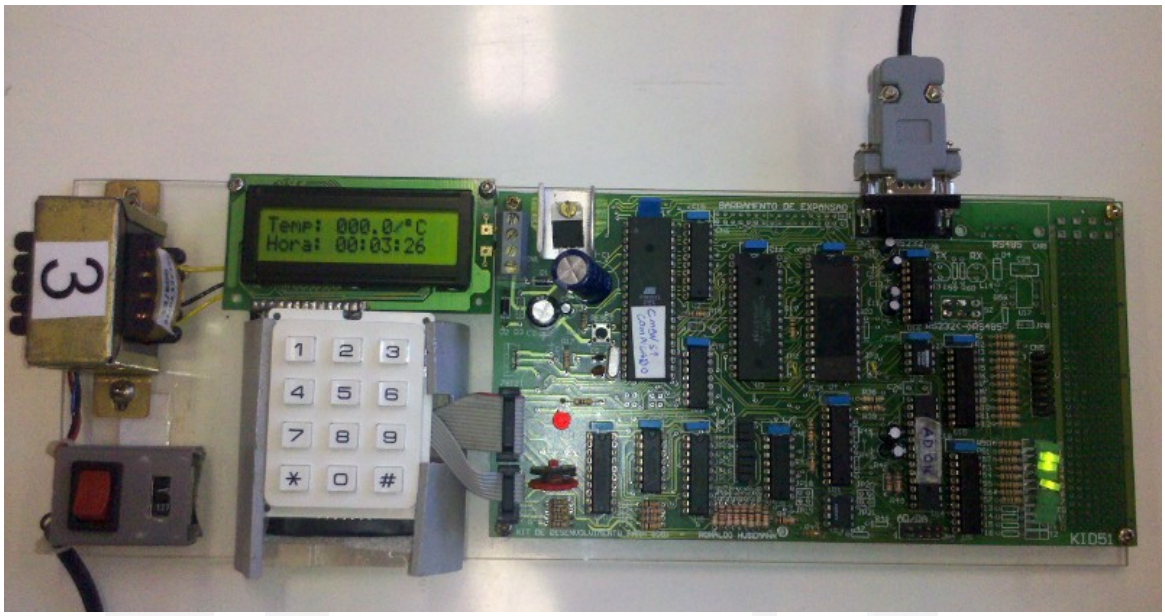
Figura 25 - Tela adquirida da ferramenta desenvolvida durante validação



Fonte: Elaborado pelo autor.

Utilizando esta ferramenta proposta, o programa desenvolvido pode ser baixado no kit de desenvolvimento para visualização dos resultados direto na placa. A Figura 26 exibe o kit em funcionamento com o software descrito.

Figura 26 - Teste experimental no kit de desenvolvimento



Fonte: Elaborado pelo autor.

Durante estes experimentos, notou-se que o comportamento do ambiente utilizando o kit e seus periféricos foi satisfatório, o que demostra a possibilidade de seu uso em ambiente acadêmico inserindo todos os benefícios que uma ferramenta como esta tem a oferecer no ensino de disciplinas de microcontroladores.

8 CONCLUSÃO

Uma IDE de programação e depuração em mesmo ambiente traz agilidade na escrita de programas e na detecção de erros no mesmo, sem a necessidade de manipular e transferir arquivos entre diferentes softwares ao mesmo tempo em que permite visualizar o estado de variáveis e memórias durante a depuração.

A possibilidade de ligação do código desenvolvido com uma placa de desenvolvimento prática permite a execução do programa em tempo real, com visualização do resultado do software em uma execução real.

Pelo fato da ferramenta desenvolvida se basear em softwares de código aberto, a atual solução não possui limitações quanto a tamanho de código ou de funcionalidades.

Baseado nas vantagens citadas, conclui-se que a ferramenta desenvolvida se apresenta como uma importante solução para facilitar o ensino técnico e universitário em áreas como programação de sistemas embarcados e arquiteturas de microprocessadores.

A contribuição para a comunidade foi a geração de um software de código fonte aberto que integra compilador e ferramentas com o usuário. Trabalhos futuros podem focar na inclusão de um simulador do hardware para experimentos onde não haja kit disponível, melhora do algoritmo de passo-a-passo, pois este se tornou lento devido sua complexidade e inclusão de suporte a outras plataformas como exemplo o Arduino.

REFERÊNCIAS

ATMEL. **AT89S52:** 8-bit Microcontroller with 8K Bytes In-System Programmable. San Rose: Califórnia. 2008. 38 p. Disponível em: <http://www.atmel.com/Images/doc1919.pdf> Acesso em: outubro de 2012.

FRAGA, C. J. **CMON51**. S. L. 2005. 11 p. Disponível em: <http://cmon51.sourceforge.net/> Acesso em: outubro de 2012.

GIMENEZ, S. P. **Microcontroladores 8051** São Paulo: Pearson Prentice Hall. 1 e. 2002. 253 p. ISBN 85-87918-28-1.

JAVA, **Java Communications API**. S. L. 2013. 1 p. Disponível em: <http://www.oracle.com/technetwork/java/index-jsp-141752.html> Acesso em: Junho de 2013a.

JAVA, **Learn About Java Technology**. S. L. 2013. 1 p. Disponível em: <http://www.java.com/en/about/> Acesso em: Junho de 2013b.

LUCAS, J. **Tutorial simples de como utilizar a ferramenta MCU 8051 ide para fazer simulações com os periféricos do kit microgenios**. S. L. 2012. 2 p. Disponível em: <http://pt.scribd.com/doc/97024660/Tutorial-MCU-8051-IDE> Acesso em: outubro de 2012.

MONTEIRO, Roberto L. S. **Tcl/tk Guia de Consulta Rápida** São Paulo: Editora Novatec. 1 e. 2001. 128 p. ISBN 85-7522-008-X.

NICOLOSI, D. E. C. **Microcontrolador 8051 Detalhado** São Paulo: Editora Érica. 4 e. 2000. 227 p. ISBN 85-7194-721-X.

NICOLOSI, D. E. C.; BRONZERI, R. B. **Microcontrolador 8051 com Linguagem C** São Paulo: Editora Érica. 2 e. 2005. 223 p. ISBN 978-85-365-0079-9.

SILVA JÚNIOR, V. P. **Aplicações práticas do microcontrolador 8051:** Teoria Geral Detalhada São Paulo: Editora Érica. 11 e. 2003. 243 p. ISBN 85-7194-939-5.

STORY, Lenny. **CDB File Format**. S. L. 2004. 14 p. Disponível em: <http://gse.ufsc.br/~bezerra/disciplinas/Microprocessadores/8051/sdcc_e_asm/sdcc/doc/cdbfileformat.pdf> Acesso em: Junho de 2012.

ANEXO A:

Instruções Assembly para a família de microcontroladores MCS51.

ANEXO: INSTRUÇÕES ASSEMBLY DA FAMÍLIA MCS51

Instruções Aritméticas

Mnemônico	Função	Tam.	Ciclos	CY	AC	OV
ADD A,Rn	Adiciona conteúdo do registrador Rn ao acumulador.	1 byte	1	x	x	x
ADD A,direto	Adiciona o conteúdo do endereço direto ao acumulador	2 bytes	1	x	x	x
ADD A,@Ri	Adiciona o conteúdo do Ri ao acumulador	1 byte	1	x	x	x
ADD A,#dado	Adiciona o dado imediato ao acumulador	2 bytes	1	x	x	x
ADDC A,Rn	Adiciona conteúdo do registrador Rn ao acumulador com carry.	1 byte	1	x	x	x
ADDC A,direto	Adiciona o conteúdo do endereço direto ao acumulador com carry	2 bytes	1	x	x	x
ADDC A,@Ri	Adiciona o conteúdo do Ri ao acumulador com carry	1 byte	1	x	x	x
ADDC A,#dado	Adiciona o dado imediato ao acumulador com carry	2 bytes	1	x	x	x
SUBB A,Rn	Subtrai do acumulador o conteúdo do registrador Rn com empréstimo	1 byte	1	x	x	x
SUBB A,direto	Subtrai do acumulador o conteúdo do endereço direto com empréstimo	2 bytes	1	x	x	x
SUBB A,@Ri	Subtrai do acumulador o conteúdo do Ri com empréstimo	1 byte	1	x	x	x
SUBB A,#dado	Subtrai do acumulador o dado imediato com empréstimo	2 bytes	1	x	x	x
INC A	Incrementa Acumulador	1 byte	1	-	-	-
INC Rn	Incrementa conteúdo do registrador Rn	1 byte	1	-	-	-
INC Direto	Incrementa o conteúdo do endereço direto	2 bytes	1	-	-	-
INC @Ri	Incrementa o conteúdo do endereço em Ri	1 byte	1	-	-	-
DEC A	Decrementa Acumulador	1 byte	1	-	-	-
DEC Rn	Decrementa conteúdo do Registrador Rn	1 byte	1	-	-	-
DEC Direto	Decrementa o conteúdo do endereço direto	2 bytes	1	-	-	-
DEC @Ri	Decrementa o conteúdo do endereço em Ri	1 byte	1	-	-	-
INC DPTR	Incrementa o ponteiro de dados	1 byte	2	-	-	-
MUL AB	Multiplica A & B. Resultado LSB em A e MSB em B	1 byte	4	-	-	-
DIV AB	Divide A por B. Resultado inteiro no A e resto no B	1 byte	4	x	-	x
DA A	Ajusta o acumulador para operação BCD	1 byte	1	x	-	x

Fonte: Adaptado de Microcontrolador 8051 Detalhado, página 165.

Instruções Lógicas

Mnemônico	Função	Tam.	Ciclos	CY	AC	OV
ANL A,Rn	E Lógico do A com conteúdo de Rn	1 byte	1	-	-	-
ANL A,direto	E Lógico do A com conteúdo endereço Direto	2 bytes	1	-	-	-
ANL A,@Ri	E Lógico do A com conteúdo do endereço Em Ri	1 byte	1	-	-	-
ANL A,#dado	E Lógico do A com dado imediato	2 bytes	1	-	-	-
ANL direto,A	E Lógico do conteúdo do endereço direto com A. Resultado no endereço direto	2 bytes	1	-	-	-
ANL direto,#dado	E Lógico do conteúdo do endereço direto com o dado imediato. Resultado no endereço direto	3 bytes	2	-	-	-
ORL A,Rn	OU Lógico do A com conteúdo de Rn	1 byte	1	-	-	-
ORL A,direto	OU Lógico do A com conteúdo endereço direto	2 bytes	1	-	-	-
ORL A,@Ri	OU Lógico do A com conteúdo do endereço Em Ri	1 byte	1	-	-	-
ORL A,#dado	OU Lógico do A com dado imediato	2 bytes	1	-	-	-
ORL direto,A	OU Lógico do conteúdo do endereço direto com A. Resultado no endereço direto	2 bytes	1	-	-	-
ORL direto,#dado	OU Lógico do conteúdo do endereço direto com o dado imediato. Resultado no endereço direto	3 bytes	2	-	-	-
XRL A,Rn	Ou-exclusivo lógico do A com conteúdo de Rn	1 byte	1	-	-	-
XRL A,direto	Ou-exclusivo lógico do A com conteúdo endereço direto	2 bytes	1	-	-	-

Mnemônico	Função	Tam.	Ciclos	CY	AC	OV
XRL A,@Ri	Ou-exclusivo lógico do A com conteúdo do endereço em Ri	1 byte	1	-	-	-
XRL A,#dado	Ou-exclusivo lógico do A com dado imediato	2 bytes	1	-	-	-
XRL direto,A	Ou-exclusivo lógico do conteúdo do endereço direto com A. Resultado no endereço direto	2 bytes	1	-	-	-
XRL direto,#dado	Ou-exclusivo lógico do conteúdo do endereço direto com o dado imediato. Resultado no endereço direto	3 bytes	2	-	-	-
CLR A	Limpa A	1 byte	1	-	-	-
CPL A	Complementa A	1 byte	1	-	-	-
RL A	Desloca A para esquerda 1 bit	1 byte	1	-	-	-
RLC A	Desloca A para esquerda 1 bit através do C	1 byte	1	x	-	-
RR A	Desloca A para direita 1 bit	1 byte	1	-	-	-
RRC A	Desloca A para direita 1 bit através do C	1 byte	1	x	-	-
SWAP A	Troca 4 bits LSB do A pelo MSB do A	1 byte	1	-	-	-

Fonte: Adaptado de Microcontrolador 8051 Detalhado, página 165.

Instruções de Transferência de Dados

Mnemônico	Função	Tam.	Ciclos	CY	AC	OV
MOV A,Rn	Copia conteúdo de Rn em A	1 byte	1	-	-	-
MOV A,direto	Copia conteúdo do endereço direto em A	2 bytes	1	-	-	-
MOV A,@Ri	Copia conteúdo do endereço em Ri no A	1 byte	1	-	-	-
MOV A,#dado	Carrega A com dado	2 bytes	1	-	-	-
MOV Rn,A	Copia conteúdo de A em Rn	1 byte	1	-	-	-
MOV Rn,direto	Copia conteúdo do endereço direto em Rn	2 bytes	2	-	-	-
MOV Rn,#dado	Carrega Rn com dado	2 bytes	1	-	-	-
MOV Direto,A	Copia conteúdo de A em endereço direto	2 bytes	1	-	-	-
MOV Direto,Rn	Copia conteúdo de Rn em endereço Direto	2 bytes	2	-	-	-
MOV Direto1,direto2	Copia conteúdo de endereço direto 2 em endereço direto1	3 bytes	2	-	-	-
MOV Direto,@Ri	Copia conteúdo do endereço em Ri no endereço direto	2 bytes	2	-	-	-
MOV Direto, #dado	Carrega dado no endereço direto	3 bytes	2	-	-	-
MOV @Ri,A	Copia conteúdo de A no endereço em Ri	1 byte	1	-	-	-
MOV @Ri,direto	Copia conteúdo do endereço direto no endereço em Ri	2 bytes	2	-	-	-
MOV @Ri,#dado	Carrega dado no endereço em Ri	2 bytes	1	-	-	-
MOV DPTR, #dado16	Carrega dado de 16 bit no DPTR	3 bytes	2	-	-	-
MOVC A,@A+DPTR	Copia conteúdo do endereço código A+DPTR em A	1 byte	2	-	-	-
MOVC A,@A+PC	Copia conteúdo do endereço código A+PC em A	1 byte	2	-	-	-
MOVX A,@Ri	Copia conteúdo do endereço dado externo em Ri no A	1 byte	2	-	-	-
MOVX A,@DPTR	Copia conteúdo do endereço dado externo em DPTR no A	1 byte	2	-	-	-
MOVX @Ri,A	Copia conteúdo de A no endereço dado externo em Ri	1 byte	2	-	-	-
MOVX @DPTR,A	Copia conteúdo de A no endereço dado externo em DPTR	1 byte	2	-	-	-
PUSH Direto	Copia conteúdo do endereço direto na próxima posição disponível da pilha	2 bytes	2	-	-	-
POP Direto	Copia conteúdo da posição corrente da pilha em endereço direto	2 bytes	2	-	-	-
XCH A,Rn	Troca conteúdo entre A e Rn	1 byte	1	-	-	-
XCH A,direto	Troca conteúdo entre A e o endereço direto	2 bytes	1	-	-	-
XCH A,@Ri	Troca conteúdo entre o A e o endereço em Ri	1 byte	1	-	-	-
XCHD A,@Ri	Troca os 4 LSB do acumulador e o endereço	1 byte	1	-	-	-

Mnemônico	Função	Tam.	Ciclos	CY	AC	OV
	em Ri					

Fonte: Adaptado de Microcontrolador 8051 Detalhado, página 165.

Instruções Booleanas

Mnemônico	Função	Tam.	Ciclos	CY	AC	OV
CLR C	Limpa C para zero	1 byte	1	x	-	-
CLR Bit	Limpa o bit indicado	2 bytes	1	-	-	-
SETB C	Coloca C em um	1 byte	1	x	-	-
SETB bit	Coloca o bit indicado em um	2 bytes	1	-	-	-
CPL C	Complementa o C	1 byte	1	x	-	-
CPL Bit	Complementa o bit	2 bytes	1	-	-	-
ANL C,bit	E entre C e o conteúdo do bit. Resultado em C	2 bytes	2	x	-	-
ANL C, /bit	E entre C e o complemento do bit. Resultado em C	2 bytes	2	x	-	-
ORL C,bit	Ou entre C e o conteúdo do bit. Resultado em C	2 bytes	2	x	-	-
ORL C, /bit	Ou entre C e o complemento do bit. Resultado em C	2 bytes	2	x	-	-
MOV C, bit	Copia conteúdo do bit em C	2 bytes	1	x	-	-
MOV Bit,C	Copia o conteúdo de C em bit	2 bytes	2	-	-	-

Fonte: Adaptado de Microcontrolador 8051 Detalhado, página 165.

Instruções de Desvio

Mnemônico	Função	Tam.	Ciclos	CY	AC	OV
JC rel	Desvie para endereço código PC+rel se C=1	2 bytes	2	-	-	-
JNC rel	Desvie para endereço código PC+rel se C=0	2 bytes	2	-	-	-
JB Bit,rel	Desvie para endereço código PC+rel se bit = 1	3 bytes	2	-	-	-
JNB Bit,rel	Desvie para endereço código PC+rel se bit = 0	3 bytes	2	-	-	-
JBC Bit,rel	Desvie para endereço código PC+rel se bit = 1 e limpa conteúdo de bit	3 bytes	2	-	-	-
ACALL ender11	Chame a subrotina em endereço absoluto PC_{15-11}, ender11. Endereço de retorno na Pilha	2 bytes	2	-	-	-
LCALL ender16	Chame a sub-rotina em ender16. Endereço retorno na Pilha	3 bytes	2	-	-	-
RET	Retorne de sub-rotina	1 byte	2	-	-	-
RETI	Retorne de rotina de interrupção	1 byte	2	-	-	-
AJMP ender11	Desvie incondicional para PC_{15-11}, ender11	2 bytes	2	-	-	-
LJMP ender16	Desvie incondicional para ender16	3 bytes	2	-	-	-
SJMP rel	Desvie Incondicional para PC+rel	2 bytes	2	-	-	-
JMP @A+DPTR	Desvie incondicional para A+DPTR	1 byte	2	-	-	-
JZ rel	Desvie para endereço PC+rel se A=0	2 bytes	2	-	-	-
JNZ rel	Desvie para endereço PC+rel se A não é zero	2 bytes	2	-	-	-
CJNE A,direto,rel	Compare e desvie para PC+rel se conteúdo de A não é igual a conteúdo do endereço direto	3 bytes	2	x	-	-
CJNE A,#dado,rel	Compare e desvie para PC+rel se dado não é igual ao conteúdo de A	3 bytes	2	x	-	-
CJNE Rn,#dado,rel	Compare e desvie para PC+rel se dado não é igual a conteúdo de Rn	3 bytes	2	x	-	-
CJNE @Ri,#dado,rel	Compare e desvie para PC+rel se dado não é igual a conteúdo do ender. em Ri	3 bytes	2	x	-	-
DJNZ Rn,rel	Decremente conteúdo de Rn e desvie para PC+rel se o novo valor em Rn não for igual a 0	2 bytes	2	-	-	-
DJNZ Direto, rel	Decremente conteúdo do endereço direto e desvie para PC+rel se o novo valor em endereço direto não for igual a 0	2 bytes	2	-	-	-

Mnemônico	Função	Tam.	Ciclos	CY	AC	OV
NOP	Sem operação	1 byte	1	-	-	-

Fonte: Adaptado de Microcontrolador 8051 Detalhado, página 165.