UNICEUB – CENTRO UNIVERSITÁRIO DE BRASÍLIA
FATECS – FACULDADE DE TECNOLOGIA E CIÊNCIAS SOCIAIS APLICADAS
CURSO DE ENGENHARIA DE COMPUTAÇÃO

Bruno de Santana Ferreira

Ciclocomputador utilizando Microcontrolador 8051 e o Compêndio de Atividades Físicas

BRASÍLIA/DF
1º SEMESTRE DE 2008

Bruno de Santana Ferreira

Ciclocomputador utilizando Microcontrolador 8051 e o Compêndio de Atividades Físicas

Monografia apresentada ao Curso de Engenharia de Computação, como requisito parcial para obtenção do grau de Engenheiro de Computação. Orientador: Prof. Thiago de Miranda Leão Toribio.

BRASÍLIA/DF
1º SEMESTRE DE 2008

Resumo

Este projeto tem como objetivo o desenvolvimento de um protótipo de ciclocomputador que retorne ao usuário o valor da perda calórica tendo como dados a sua massa corporal e o tempo de exercício, utilizando o equivalente metabólico (MET) encontrado no Compêndio de Atividades Físicas. Além de informar a distância percorrida e velocidade.

O ciclocomputador será desenvolvido com o microcontrolador AT89C2051, e a sua programação em linguagem *Assembly*. Os circuitos serão feitos utilizando *protoboard* e os componentes necessários para posterior construção de uma placa.

Palavras-Chave: Ciclocomputador, Microcontrolador, AT89C2051, Compêndio de Atividades Físicas, Perda Calórica, Equivalente Metabólico.

Abstract

The objective of this project is the development of a cyclocomputer prototype which can return to the user a certain value of caloric loss according to his body mass and the duration of the exercise, using the metabolic equivalent (MET) found in the Compendium of Physical Activities. Besides indicating the value of caloric loss, the prototype will also inform the distance covered and the velocity.

The cyclocomputer will be developed with a microcontroller AT89C2051, and its programming will be done in the Assembly language. The circuit will be made with a protoboard and the electronic components for the further construction of the board.

Wordkeys: Cyclocomputer, Microocontroller, AT89C2051, Compendium of Physical Activities, Calory Loss, Metabolic Equivalent

Agradecimentos

Aos meus pais, pelo apoio e incentivo;

À minha namorada, Ângela, pelo amor, carinho e por me ajudar a enfrentar esta etapa da minha vida;

Aos meus familiares que participaram e me apoiaram durante toda essa jornada;

E, finalmente, agradeço ao meu orientador Thiago Toribio por me direcionar de forma correta e por sua competência e disponibilidade.

Sumário

Índice de Figuras

Índice de Tabelas

1. INTRODUÇÃO

De acordo com a fabricante de bicicletas Caloi, a primeira bicicleta foi construída em 1790, pelo conde francês *Méde de Sivrac*, sendo nomeada de "celerífero". Desde então a bicicleta teve grandes evoluções, deixando de ser uma iguaria dos nobres e tornando-se um meio de transporte popular de fácil utilização, além de ser indicada na 57ª Assembléia Mundial de Saúde (2004), como uma das saídas para melhorar a qualidade de vida mundial.

Segundo a Associação Brasileira dos Fabricantes de Motocicletas, Ciclomotores, Motonetas, Bicicletas e Similares, ABRACICLO, no mundo foram produzidas e consumidas cerca de 120 milhões de bicicletas em 2002, e deste número, foram produzidas e consumidas no Brasil 5 milhões, tornando-o o 3° maior produtor e o 5° maior consumidor de bicicletas do mundo.

Tabela 1.1: Produção Mundial de Bicicletas 2002

Posição	País	Unidades (em Milhões)	%
1	China	80.0	66.7
2	Índia	10.0	8.3
3	Brasil	5.0	4.2
4	Alemanha	3.0	2.5
5	Japão	3.0	2.5
6	Taiwan	2.5	2.1
7	Itália	2.5	2.1
8	Vietnam	2.0	1.7
9	França	2.0	1.7
10	Holanda	1.0	0.8
Outros		9.0	7.5
Total		**120.0**	**100.0**

Fonte: Bike Europe, Bicycle Retailer and Industry News (BRAIN); NBDA (National Bicycle Dealer Association), disponível em: <http://www.abraciclo.com.br/prodbic.html>.

Tabela 1.2: Consumo Mundial de Bicicletas 2002

Posição	País	Unidades (em Milhões)	%
1	China	35.0	29.2
2	Estados Unidos	19.7	16.4
3	Japão	11.0	9.2
4	Índia	10.0	8.3
5	Brasil	5.0	4.2
6	Alemanha	4.6	3.8
7	França	3.0	2.5
8	Inglaterra	2.4	2.0
9	Itália	1.4	1.2
10	Holanda	1.3	1.1
Outros		26.6	22.1
Total		**120.0**	**100.0**

Fonte: Bike Europe, Bicycle Retailer and Industry News (BRAIN); NBDA (National Bicycle Dealer Association), disponível em: <http://www.abraciclo.com.br/consumobic.html>.

Estimativas da ABRACICLO (2004) eram de uma frota brasileira de 60 milhões de bicicletas, com 18% desse total destinado para lazer e esportes.

Tabela 1.3: Segmentação do Mercado

Tipo	Participação
Transporte	53%
Infantil	29%
Lazer	17%
Esporte	1%

Fonte: ABRACICLO, disponível em: http://www.abraciclo.com.br/

Com a sua popularização, tanto para fins de transporte quanto para lazer, os usuários tiveram a necessidade de obter dados sobre a utilização da bicicleta. Parâmetros como quantos quilômetros o usuário percorria em seu trajeto, a velocidade desenvolvida e o tempo gasto no percurso começaram a ser demandados pelos ciclistas. Equipamentos de medidas começaram a

ser instalados nas bicicletas: odômetros para a medição das distâncias percorridas, velocímetros para velocidade, cronômetros para o tempo gasto, etc.

Novas demandas de informações, como o consumo calórico, a freqüência cardíaca e cadência (pedaladas por minuto), trouxeram a necessidade de condensar os vários equipamentos responsáveis por tais medidas em um equipamento mais simples e compacto, que pode ser levado na bicicleta sem comprometer o seu uso. A partir dessa necessidade foi criado o ciclocomputador.

1.1. MOTIVAÇÃO

Para obter os dados e gerar tais informações, o ciclocomputador utiliza sensores. Como exemplo pode-se citar o sensor magnético, que obtém o número de voltas da roda da bicicleta, e o sensor de freqüência cardíaca, que informa os batimentos cardíacos ao ciclista.

Nos ciclocomputadores não existem sensores ou outra forma de obter algum dado físico do usuário, apesar de alguns possuírem a opção de exibir o gasto calórico, mesmo não tendo dados da fisiologia do usuário para alimentar esta informação. Então a perda calórica é calculada a partir de uma média para todos os praticantes do esporte, o que não torna a informação confiável, principalmente para aqueles que objetivam a perda de peso através do ciclismo.

1.2. OBJETIVOS

Com o objetivo de oferecer esta informação com mais precisão, o seguinte trabalho propõe o desenvolvimento de um protótipo de ciclocomputador, utilizando um microcontrolador da família 8051, que informe ao usuário a perda calórica gerada através da inserção de sua massa. Utilizando o equivalente metabólico (MET), obtido no Compêndio de Atividades Físicas, pode-se relacionar a perda calórica com o tempo de exercício, obtendo assim um valor de gasto energético mais específico para cada usuário.

Além de exibir a perda calórica utilizando as relações mencionadas, o ciclocomputador também exibirá a distância total percorrida e a velocidade do ciclista.

O projeto prevê a construção de um protótipo de natureza acadêmica. Ele será desenvolvido utilizando o microcontrolador AT89C2051, display LCD e o sensor magnético *reed-switch*, sendo primeiramente trabalhado em um *protoboard* para posterior confecção da placa de circuitos. Será utilizada a linguagem *Assembly* para a criação do algoritmo.

1.3. ESTRUTURA DA MONOGRAFIA

O presente trabalho foi dividido em cinco capítulos.

No capítulo 1 é apresentada a introdução, motivação, objetivo e estrutura da monografia.

No capítulo 2 é apresentado o referencial teórico do trabalho, envolvendo as disciplinas de fisiologia, física, matemática e microcontroladores, visando o que será utilizado para o desenvolvimento do protótipo.

No capítulo 3 é apresentado o desenvolvimento do projeto, mencionando mais especificadamente o desenvolvimento do *hardware* e do *software*.

No capítulo 4 é abordada a análise de desempenho do protótipo com os testes realizados.

No capítulo 5 são apresentadas as considerações finais, onde será relatada a conclusão da monografia, apresentação dos resultados finais, dificuldades encontradas e propostas de trabalhos futuros.

2. REFERENCIAL TEÓRICO

Neste capítulo são abordadas as principais teorias utilizadas no projeto. Na seção sobre aspectos matemáticos e físicos são abordadas as equações de distância e velocidade média, além do cálculo da circunferência da roda da bicicleta. Na seção sobre fisiologia, são explicados os conceitos de equivalente metabólico (MET), o Compêndio de Atividades Físicas e a equação utilizada para o cálculo da perda calórica. E na última seção, sobre microcontrolador, é tratada toda a teoria utilizada no ciclocomputador.

2.1. ASPECTOS MATEMÁTICOS E FÍSICOS

Esta seção tem como objetivo explicar a origem, definições e equações da distância percorrida e da velocidade, além das equações de trabalho e potência, necessários para a compreensão da seção de fisiologia.

2.1.1. ASPECTOS MATEMÁTICOS

A distância é definida comumente como o intervalo espacial de separação entre dois pontos. Para o projeto ela será o intervalo que separa a primeira ativação do sensor e a última.

Para calculá-la são necessárias duas informações: o perímetro da circunferência do conjunto roda e pneu da bicicleta e o número de voltas que roda realizou.

Pela geometria obtêm-se o cálculo do perímetro da circunferência, que tem dependência com o raio do conjunto roda e pneu. Esta equação é:

$$\textbf{Perímetro da Circunferência} = 2.\pi.r \qquad \textbf{(2.1)}$$

No protótipo, o perímetro da roda será fixo, específico para bicicletas de aro 26”, que no caso é de 2,05m.

FIGURA 2.1: Dados do conjunto roda-pneu

O número de voltas da roda é obtido através do sensor instalado na forquilha superior. Dois ímãs, colocados nesta roda a 180° um do outro, acionam o sensor toda vez que passarem por ele, e essa informação é armazenada por um contador no microcontrolador. Sempre que for atualizado, esse valor é multiplicado pela metade do perímetro da circunferência do conjunto, conseguindo assim a exibição da distância percorrida metro a metro.

2.1.2. ASPECTOS FÍSICOS

Aqui são tratadas a obtenção da velocidade e a relação entre trabalho, energia e potência.

2.1.2.1. VELOCIDADE

Em um percurso uma bicicleta pode ir em linha reta, fazer curvas, subir e descer elevações. Para calcular a velocidade, podemos simplificar todas essas variações de percurso para simplesmente a distância percorrida, transformando a realidade em um modelo unidimensional.

O movimento unidimensional pode ser ilustrado como o "movimento de uma partícula ao longo de uma reta." (Tipler, 2000). Ou como o modelo apresentado para este trabalho, uma bicicleta em percurso reto.

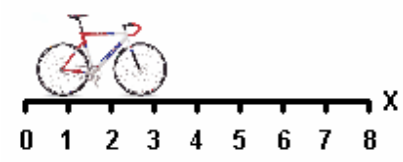


FIGURA 2.2: Deslocamento Unidimensional

A velocidade média é "a razão do deslocamento ao intervalo de tempo que ele leva para se produzir." (Nussenzveig, 1992, p. 44), ou seja, para o cálculo da velocidade média, dois fatores devem ser apresentados: o deslocamento e o intervalo de tempo.

$$v = \frac{\Delta x}{\Delta t} \quad \textbf{(2.2)}$$

O deslocamento, ou Δx, é a distância percorrida pela bicicleta, medida em metros (m), deste o ponto de partida até o ponto onde quer se verificar a velocidade.

O tempo, Δt, é o intervalo de tempo, em segundos (s), que levou para percorrer o deslocamento. O tempo será calculado a partir do ciclo de máquina do microcontrolador. Esse cálculo será abordado mais adiante, na seção sobre microcontroladores.

Fixando o tempo da velocidade em 1 segundo e utilizando um contador para contar a distância percorrida neste tempo, obtém-se a velocidade em metros por segundo (m/s) que o ciclista realizou naquele segundo de deslocamento, podendo assim ser considerada a velocidade instantânea.

Para conveniência do usuário, a velocidade é multiplicada por "3,6", passando assim para quilômetros por hora (km/h), unidade de velocidade em que ele tem maior contato. Como a contagem realizada pelo protótipo é feita com o número de metros percorridos em 1 segundo, para posteriormente haver a conversão, o resultado final acaba sendo um valor múltiplo de 3,6.

2.1.2.2. TRABALHO, ENERGIA E POTÊNCIA

Ao pedalar a bicicleta, o ciclista aplica uma força sobre ela e Tipler (2000) conceitua trabalho como uma força agindo sobre um corpo, havendo assim deslocamento. Intrinsecamente ligado ao trabalho, a energia é a "capacidade de produzir trabalho" (Nussenzveig, 1992, p. 176).

Pelas duas definições, chega-se a conclusão de que trabalho e energia são grandezas correlatas, já que o primeiro é o gasto do segundo.

Devido a essa correlação, ambos são medidos através da mesma unidade no Sistema Internacional de Unidades (SI), o joule (J). Mas por ser um dos conceitos fundamentais da física, energia é encontrada em várias de suas áreas, como termodinâmica, eletromagnetismo e mecânica, citada acima.

Na termodinâmica, segundo Tipler (2000), é aceito que o calor, definido usualmente por caloria (cal), é uma forma de energia; portanto pode ser escrito em termos de unidades adotadas pelo SI, ou seja, através do joule. A relação entre joule e a caloria é chamada de equivalente mecânico de calor:

$$\mathbf{1\ cal = 4{,}184\ J} \qquad \mathbf{(2.3)}$$

É importante ressaltar que as ditas calorias obtidas através da alimentação e perdidas através de atividades físicas são na verdade quilocalorias (kcal).

A potência é a "taxa temporal com que a força efetua um trabalho" (Tipler, 2000, p. 149), ou seja, é o trabalho avaliado pelo tempo. Pelo SI, sua unidade é o J/s ou watt (W). Ela é importante, pois permite avaliar o desempenho na aplicação do trabalho.

Voltando ao cenário do ciclista pedalando sua bicicleta, caso ele aplique um trabalho no tempo t_1 e depois volte a aplicar o mesmo trabalho, mas em um tempo t_2, sendo que $t_2 > t_1$,

demonstra que ele obteve maior potência durante a primeira etapa, já que gastou menos tempo para aplicar a mesma quantidade de trabalho, assim obtendo melhor desempenho.

2.2. FISIOLOGIA

> Quando nos exercitamos, nossos corpos usam energia química derivada do catabolismo para produzir a contração muscular. Durante esse processo usamos energia e, conseqüentemente, gastamos calorias e produzimos potência mecânica de trabalho. Para melhor entender o processo de potência, de trabalho e de gasto energético durante o exercício, cientistas desenvolveram métodos que podem quantificar essas capacidades. (Robergs; Roberts, 2002, p. 55)

A calorimetria é um dos métodos com o qual se pode quantificar o gasto energético, ou gasto calórico, já que "o calor é o principal subproduto das reações químicas do corpo" e "a medição do calor produzido pelo corpo seria um reflexo da taxa de metabolismo corporal." (Robergs; Roberts, 2002, p. 61). Calorimetria, "A ciência que quantifica a liberação de calor do metabolismo" (Robergs; Roberts, 2002, p. 61), pode ser mensurada de duas formas: diretamente e indiretamente.

A forma direta envolve "a medição direta de dissipação do calor" utilizando um calorímetro e a indireta "é calculada por outras medidas" (Robergs; Roberts, 2002, p. 61), como, por exemplo, o consumo de oxigênio.

"Muitos cientistas demonstraram que a quantidade de O_2 consumida em repouso ou ao realizar um trabalho, quando enunciada em equivalentes calóricos (kcal), será igual ao calor produzido pelo corpo, conforme determinado diretamente em um calorímetro" (Foss; Keteyian, 2000, p. 73). Portanto o volume de O_2 consumido (VO_2) torna-se uma medida indireta de energia.

Como explicado no subitem “Aspectos Físicos”, a energia pode ser enunciada em unidades além do Joule (J), como caloria (cal) ou quilocaloria (kcal), obviamente respeitando a relação existente entre as mesmas. Portanto, quando o volume de O_2 (VO_2) também for enunciado em equivalente calórico (kcal), o VO_2 torna-se equivalente ao trabalho realizado.

Também explicado no subitem “Aspectos Físicos”, a potência é o trabalho realizado em um determinado tempo. De acordo com o Sistema Internacional de Unidades (SI), a unidade de tempo utilizada deve ser o segundo (s), mas nas medidas fisiológicas é mais interessante expressar o tempo em termos de minutos (min). Deste modo, quando avaliado o volume de O_2 consumido por minuto ($\dot{V}O_2$), sendo expresso por quilocaloria por minuto (kcal/min) , tem-se que o $\dot{V}O_2$ é equivalente à potência, no caso, a potência gasta ou o dispêndio energético.

Ainda de acordo com Foss e Keteyian (2000), o consumo de oxigênio por minuto de uma pessoa em repouso define outra variável, o equivalente metabólico (MET). “Um MET é definido como o dispêndio energético ($\dot{V}O_2$) enunciado como um ml/kg/min (ou l/min) (*sic*) em condições de repouso tranqüilo” (Foss; Keteyian, 2000, p. 86)

Ou seja, se um exercício requer 5 METs, significa que o consumo de O_2 por minuto é cinco vezes maior que o consumo quando em repouso.

“Para um adulto comum, 1 MET é de aproximadamente 3,5 ml de O_2 consumido por quilograma de peso corporal por minuto (1 MET = 3,5 ml/kg/min). É também de aproximadamente 1 kcal/kg de peso corporal/hora” (Foss; Keteyian, 2000, p. 86), ou resumidamente, aproximadamente 1 kcal/kg/h. Amorim e Gomes (2003, p. 165) também definem

a taxa metabólica de repouso (TMR) como “muito próxima a 1 kcal.kg de peso corporal^{-1}.h^{-1}”, que também é 1 kcal/kg/h, ou seja, a taxa metabólica de repouso (TMR) é equivalente ao equivalente metabólico (MET) quando seu valor é 1.

Amorim e Gomes (2003) dizem que, se o peso corporal (em kg) de uma pessoa for multiplicado pelo valor MET do exercício que ela realizou, pelo tempo de exercício (em minutos) e dividido por 60 minutos é possível estimar a perda calórica em kcal.

Tem-se então a fórmula para perda calórica com base no valor MET do exercício, o peso do praticante e o tempo de exercício:

Gasto Energético (Kcal) = MET x massa (Kg) x tempo (min) / 60 min (2.4)

Obtêm-se valores MET para vários exercícios através do Compêndio de Atividades Físicas. De acordo com Amorim e Gomes (2003), o compêndio foi criado por Barbara Ainsworth e colaboradores, em 1989, buscando criar um sistema padronizado de codificação de atividades físicas. Tabela a seguir é mostrada a parte do Compêndio relacionado ao ciclismo.

Tabela 2.1: Compêndio de Atividades Físicas - Bicicletas

Código	METS	Atividade Específica	Exemplo
01009	8,5	Andar de bicicleta	Andar de bicicleta BMX ou de montanha
01010	4	Andar de bicicleta	Andar de bicicleta, <16 km/h, lazer, a trabalho ou prazer
01015	8	Andar de bicicleta	Andar de bicicleta, geral
01020	6	Andar de bicicleta	Andar de bicicleta, 16-19 km/h, lazer, baixa velocidade, esforço
01030	8	Andar de bicicleta	Andar de bicicleta, 19-22 km/h, lazer, esforço moderado
01040	10	Andar de bicicleta	Andar de bicicleta, 22-25 km/h, corrida ou lazer, rápido, esforço
01050	12	Andar de bicicleta	Andar de bicicleta, 25-30 km/h, correr sem acoplar ou 30 km/h
01060	16	Andar de bicicleta	Andar de bicicleta, >32 km/h, correndo sem acoplar
01070	5	Andar de bicicleta	Monociclo

Fonte: Amorim; Gomes (2003)

> Deve ser enfatizado que o compêndio foi desenvolvido para facilitar a codificação de atividades físicas e comparar estudos. Não são levadas em consideração as diferenças individuais que podem alterar o custo energético do movimento. Assim, um fator de correção pode ser necessário para ajustes individuais, o que, no entanto, não existe até o momento. (Amorim; Gomes, 2003, p. 142)

Por exemplo, uma pessoa de 80 quilogramas pedala uma bicicleta abaixo de 16 km/h durante 20 minutos. Para encontrar seu gasto energético, em kcal, basta multiplicar o valor MET do exercício (no caso 4 METs) pela sua massa (80 kg) e pelo tempo de exercício (20 min), dividindo tudo por 60 minutos. O gasto energético dessa pessoa nesta atividade foi de 106,6 kcal.

Gasto Energético = MET x massa x tempo / 60

Gasto Energético = 4 kcal/kg x 80 kg x 20 min / 60 min

Gasto Energético = 106,6 kcal

Também é citado que “a TMR não é igual a 1 kcal.kg de peso corporal^{-1} para todos os indivíduos” e que as “intensidades absolutas em MET, propostas no compêndio, podem ser inexatas para pessoas com diferentes massas corporais e percentuais de gordura.” (Amorim; Gomes, 2003, p. 165).

Segundo a revista Running BR (2006), diz que a perda calórica calculada em esteiras não é confiável, pois é estimativa vaga, e que não considera o principal fator para avaliar o gasto calórico de uma pessoa durante o exercício, o condicionamento físico. O mesmo pode ser dito com relação aos ciclocomputadores para bicicletas.

Apesar desses fatores, Amorim e Gomes (2003) concluem que esta proposta tem sido utilizada em várias regiões do planeta e que sua maior qualidade é que o compêndio vem sendo atualizado continuamente, corrigindo possíveis distorções, além de possuir várias atividades listadas.

2.3. MICROCONTROLADORES

O microcontrolador é um dos principais componentes do ciclocomputador. Ele recebe todos os dados necessários, processa-os e retorna ao usuário, pelo *display LCD*, as informações geradas. Nesta subseção são tratados os principais componentes utilizados no desenvolvimento do protótipo: o microcontrolador, desenvolvido através do microprocessador, as memórias e periféricos internos, as características da família de microcontroladores 8051 (seus pinos e método utilizado para a busca de instruções), a organização da memória RAM interna e a utilização dos temporizadores.

2.3.1. MICROPROCESSADOR

"Um microprocessador é um elemento eletrônico, desenvolvido para executar tarefas específicas, com linguagem de comando específica" (Nicolosi, 2004, p. 60). Segundo Nicolosi (2004) o microprocessador é apenas um *chip* responsável pelo processamento. Para seu funcionamento, *chips* adicionais devem ser utilizados: uma memória ROM (*Read-only Memory)* onde é gravada a programação e onde o microprocessador busca as instruções, memória RAM (*Random Access Memory*), utilizada para armazenar temporariamente as informações das instruções, barramentos para as ligações das memórias com o microprocessador, um oscilador para gerar o ciclo de máquina, periféricos para interagir com o ambiente e a fonte de alimentação.

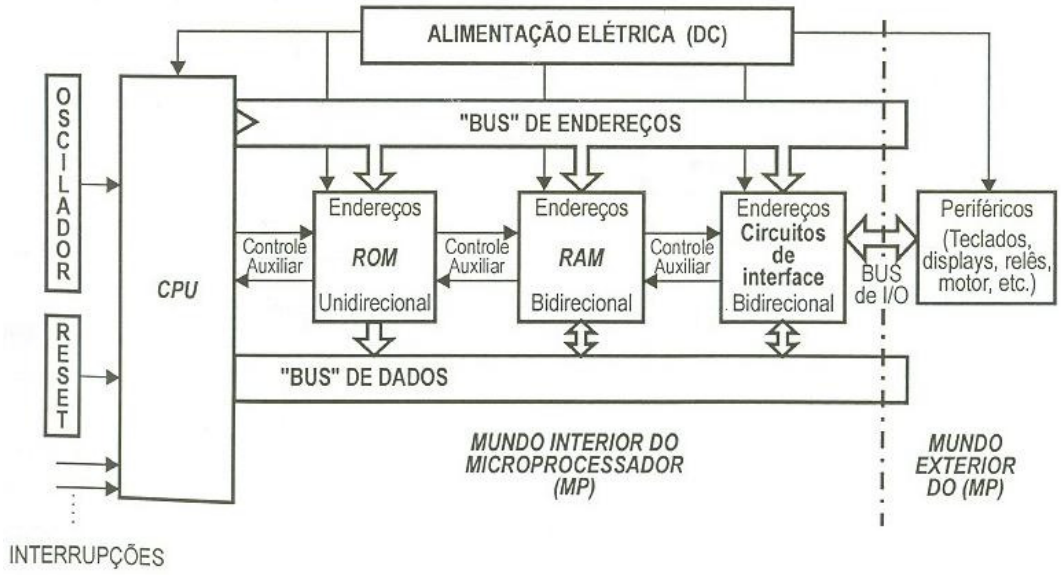


FIGURA 2.3: Arquitetura Básica do Microprocessador
Fonte: Nicolosi (2004)

De acordo com Nicolosi (2004), no interior do microprocessador contém um Contador de Programas com a função de indicar o endereço da próxima instrução a ser buscada, um Registrador de Instruções, onde a instrução buscada na ROM é armazenada, uma Unidade de

Decodificação, onde a instrução é decodificada, uma Unidade Lógica e Aritmética, onde as operações de lógica, aritmética além de decisões e comparações são realizadas, uma Unidade de Controle, local onde ocorre o processamento do controle de fluxo de informações para a realização da instrução recebida e o Acumulador, um dos principais registradores utilizados nas instruções.

Pela utilização desses componentes, o microprocessador pode realizar a busca e execução das instruções encontradas no programa gravado na ROM. Esse ciclo de busca é iniciado com o Contador de Programa informando o endereço da próxima instrução a ser buscada na via de endereços, a instrução é lida na ROM pela via de dados contida na via de endereços, essa instrução é então armazenada no Registrador de Instruções e o contador é incrementado em um, para poder indicar o próximo endereço. A execução da instrução então é iniciada, utilizando-se da Unidade de Controle e da Unidade de Decodificação e executada pela Unidade Lógica e Aritmética.

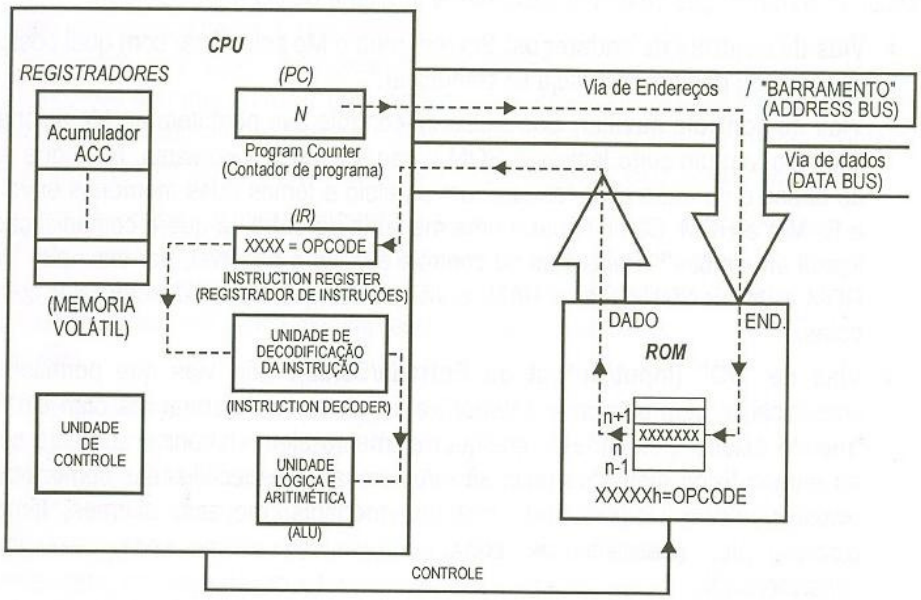


FIGURA 2.4: Ciclo de busca de instruções
Fonte: Nicolosi (2004)

2.3.2. MICROCONTROLADOR

Enquanto o microprocessador é apenas um *chip* de processamento, que utiliza *chips* adicionais como *chips* de memória para seu funcionamento, o microcontrolador "já tem estes *chips* dentro da mesma pastilha do microprocessador. O microcontrolador corresponde a um microprocessador e seus periféricos típicos, todos juntos num só *chip*" (Nicolosi, 2004, p. 65). Ou seja, em seu interior o microcontrolador possui um microprocessador, uma memória ROM e RAM, barramentos, interligações para entrada e saída, *timers* para a contagem de tempo entre outros.

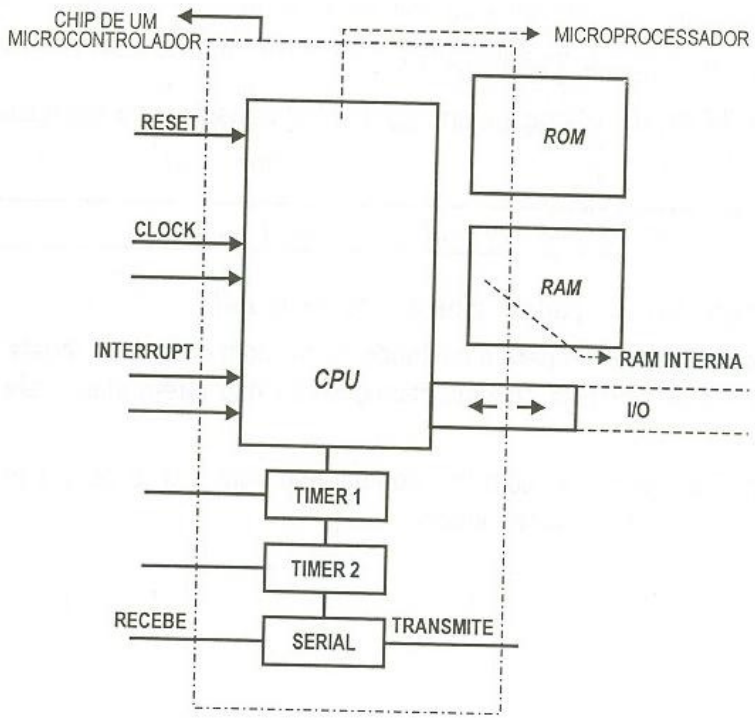


FIGURA 2.5: Diferença entre microprocessador e microcontrolador
Fonte: Nicolosi (2004)

2.3.2.1. CARACTERÍSTICAS GERAIS DA FAMÍLIA 8051

O microcontrolador utilizado no ciclocomputador, o AT89C2051, pertence à família 8051, de acordo com a ATMEL (2005). Portanto, ter uma visão geral do microcontrolador padrão da família permitirá uma melhor compreensão do microcontrolador utilizado.

Segundo Nicolosi (2004), o microcontrolador 8051 típico possui uma RAM com 128 *bytes* de uso geral e 128 *bytes* para registradores especiais, ROM interna de 4 *Kbytes*, 4 *ports* de entrada e saída, 2 *Timers* de 16 *bits* e entradas de interrupção externas.

O microcontrolador utiliza seus pinos para realizar a comunicação com periféricos, ligar-se a *chips* externos, receber a alimentação e ligar-se com o aterramento; o 8051 possui 40 pinos e suas descrições são apresentadas a seguir, de acordo com Nicolosi (2004) e Junior (1998).

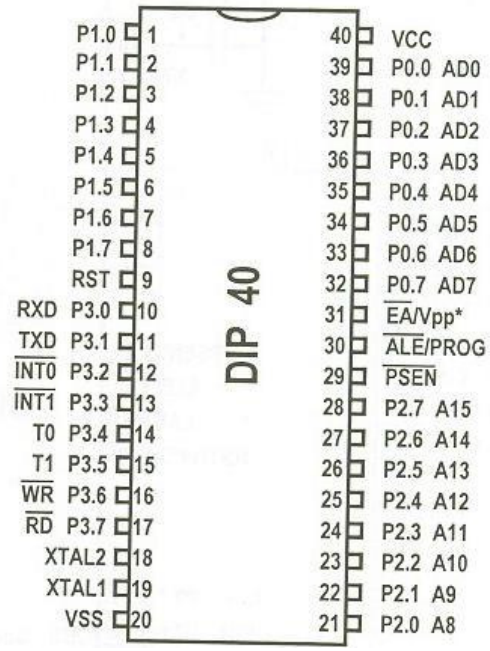


FIGURA 2.6: Organização dos pinos do 8051
Fonte: Nicolosi (2004)

- *Port* 0: Compreende os pinos P0.0 a P0.7. É usada com propósitos gerais, caso não esteja utilizando memórias externas. Caso esteja utilizando, a *port* 0 assume a função de multiplexar dados e endereços menos significativos com essas memórias.
- *Port* 1: Compreende os pinos P1.0 a P1.7. É usada para propósitos gerais, comumente ligado a dispositivos de entrada e saída.
- *Port* 2: Compreende os pinos P2.0 a P2.7. Também é uma *port* de uso geral, mas quando se é usado memórias externas, esta *port* é utilizada com o barramento de endereços, ficando responsável pela parte mais significativa do mesmo.
- *Port* 3: Compreende os pinos P3.0 a P3.7. Assim como a *port* 1, é utilizada para propósitos gerais, sendo mais comumente usada para entrada e saída. Caso seja usada interrupção externa, RAM externa ou periféricos internos, como os *timers*, os pinos da port 3 recebem as seguintes funções especiais:

Tabela 2.2: Funções especiais da port 3

Pino	Função	Descrição
P3.0	RDX, Receptor de dados	Receptor de dados serial
P3.1	TXD, Transmissor de dados	Transmissor de dados serial
P3.2	Interrupção externa 0	Interrupção por evento externo
P3.3	Interrupção externa 1	Interrupção por evento externo
P3.4	Timer/Counter 0, entrada externa	Timer 0 utilizado como contador de eventos externos
P3.5	Timer/Counter 1, entrada externa	Timer 1 utilizado como contador de eventos externos
P3.6	Escrita externa	Escrita em RAM externa
P3.7	Leitura externa	Leitura em RAM externa

- PSEN\ (*Program Store Enable*): É um pino de controle de ROM externa. Quando o microcontrolador faz uma busca por instruções nesta memória, este pino a ativa durante a busca.
- ALE (*Address Latch Enable*): Pino usado para a ativação do *latch*, responsável por demultiplexar os dados e endereços enviados pela *port* 0 quando se trabalha com memórias externas.
- EA\ (*External Access*) Pino de comando externo, usado para determinar a utilização da ROM interna ou externa.
- RST: Pino responsável por receber o sinal de *reset*, permitindo que o microcontrolador seja iniciado adequadamente. Para que o *reset* ocorra, é necessário que este pino receba o valor lógico 1 durante pelo menos 2 ciclos de máquina.
- XTAL1 e XTAL2: Pinos ligados a um cristal externo, usado pelo sistema de oscilação interno do microcontrolador para geração do *clock*. Ele é que define o ciclo de máquina do microcontrolador (ciclo de máquina = 12 x Período de *clock*) e todas as atividades internas e externas são comandadas pelo ciclo de máquina.
- VCC: Ligado a alimentação de 5V.
- VSS: Ligado ao aterramento.

2.3.2.2. ORGANIZAÇÃO DA RAM INTERNA

O microcontrolador 8051 possui 256 *bytes* de memória RAM interna. Ela é dividida em bancos de registradores, área de *bit* e *bytes* endereçáveis, área de *bytes* endereçáveis e registradores de funções especiais.

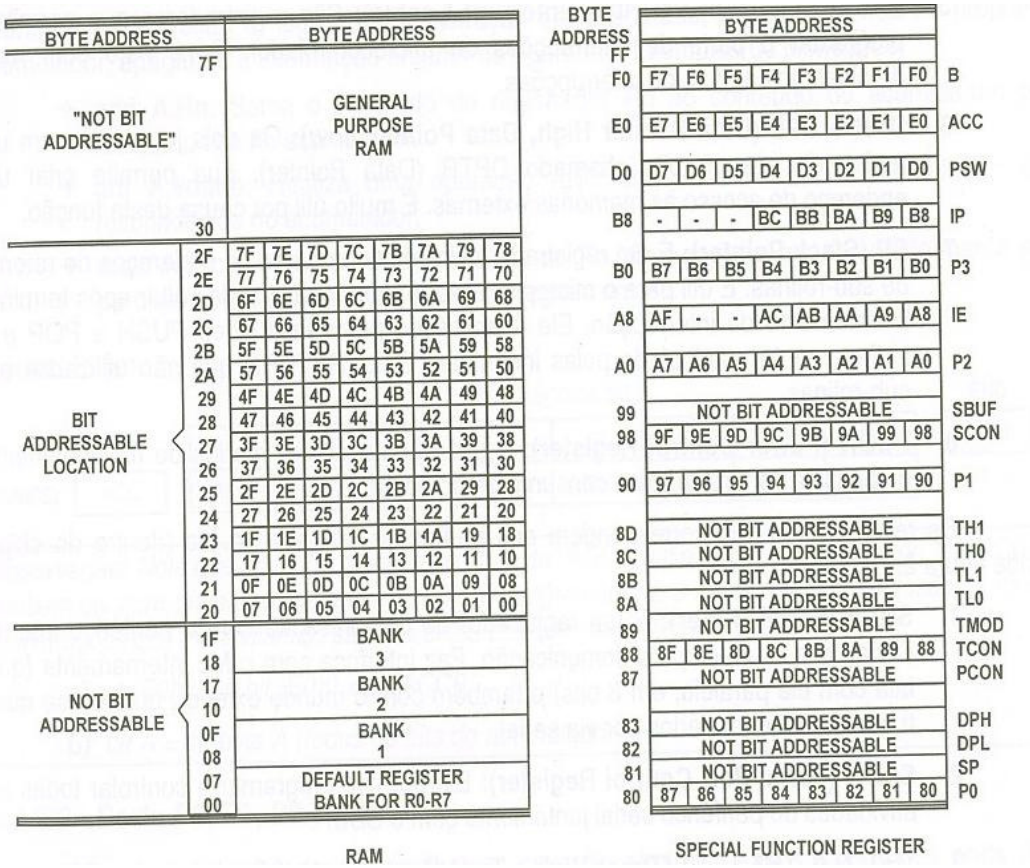


FIGURA 2.7: Mapa da RAM interna
Fonte: Nicolosi (2004)

"Os Bancos de Registradores são simples posições de memória RAM que permitem seu endereçamento pelo nome dado a cada registro, além de seu endereçamento pela posição de memória." (Junior, 1998, p. 26). São quatro bancos de registradores, banco 0 ao banco 3, cujas suas posições de memória são nomeadas de R0 a R7 em cada um deles. O microcontrolador é

iniciado com o banco 0 ativo, e para acessar os registradores dos outros bancos basta mudar de banco durante a programação.

Nicolosi (2004) explica que a área da RAM endereçável por *bit* e *byte* vem logo após os bancos de registradores. Nesta área, seus *bytes* não recebem nomes específicos, como no banco de registradores ou nos registradores de funções especiais, mas permite que sua área de memória seja chamada pelo endereço do *byte* ou de seus *bits*. Já a área da RAM endereçável apenas por *byte* permite que sua área de memória seja chamada apenas pelo endereço de seus *bytes*.

Ainda segundo Nicolosi (2004), os registradores de funções especiais são registradores que possuem funções específicas no microcontrolador. Todos são endereçáveis por *byte*, e apenas alguns permitem o endereçamento por bit.

- ACC: É o acumulador, registrador utilizado por várias instruções.
- B: Registrador utilizado em operações aritméticas de multiplicação e divisão.
- P0, P1, P2, P3: Registradores das *ports* do microcontrolador.
- PSW: Utilizado para registrar o estado da última operação lógica e aritmética feita pelo acumulador.
- IP e IE: Registradores onde são programadas as interrupções.
- DPH e DPL: São os registradores que, juntos, formam o DPTR, registrador de 16 *bits*. O DPH recebe a parte mais significativa do valor, enquanto o DPL a parte menos significativa.
- SP: Registrador utilizado para armazenar o endereço de retorno de sub-rotinas.

- PCON: Permite adaptar o microcontrolador para armazenamento de informações em caso de falha na alimentação.
- SBUF: Responsável pela comunicação com o serial.
- SCON: Utilizado junto ao SBUF para a programação e controle do serial.
- TH0, TL0, TH1, TL1: São os registradores que formam o *Timer* 0 e *Timer* 1, que, como o DPTR, são registradores de 16 *bits*.
- TCON: Controla as atividades de ambos os *Timers*.
- TMOD: Permite a programação do modo de operação dos *Timers*.

2.3.2.3. TEMPORIZADORES

"Nos sistemas microprocessados em geral, os temporizadores são utilizados para gerar periódicos e precisos pedidos de interrupção, medir largura de pulsos externos, contagem de tempo, entre outras funções." (Junior, 1998, p. 63). Os temporizadores, mais comumente conhecido como *Timers* nos microcontroladores, utilizam do ciclo de máquina, gerado pelo *clock* do microcontrolador, para gerar as contagens de tempo; fazem parte dos periféricos internos, sendo que o 8051 possui *Timer* 0 e o *Timer* 1. Para a programação dos *Timers*, são utilizados os registradores TCON e TMOD.

De acordo com Nicolosi (2004), o TCON possui seus bits divididos em duas partes, uma referente a interrupções e outra utilizada nos *Timers*. A seguir está a imagem do registrador TCON e a função dos bits relacionados aos *Timers*:

FIGURA 2.8: Registrador TCON
Fonte: Nicolosi (2004)

- TF1: *Bit* responsável por indicar o estouro da contagem no *Timer* 1. Ele é setado sempre que ocorrer um *overflow*.
- TR1: Quando setado, o *Timer* 1 é ligado, e quando resetado, a contagem é parada.
- TF0: *Bit* responsável por indicar o estouro da contagem no *Timer* 0. Ele é setado sempre que ocorrer um *overflow*.
- TR0: Quando setado, o *Timer* 0 é ligado, e quando resetado, a contagem é parada.

O TMOD, também de acordo com Nicolosi (2004) é responsável por indicar o modo de funcionamento dos *Timers*. A seguir está a imagem do registrador TMOD e a função de seus *bits*:

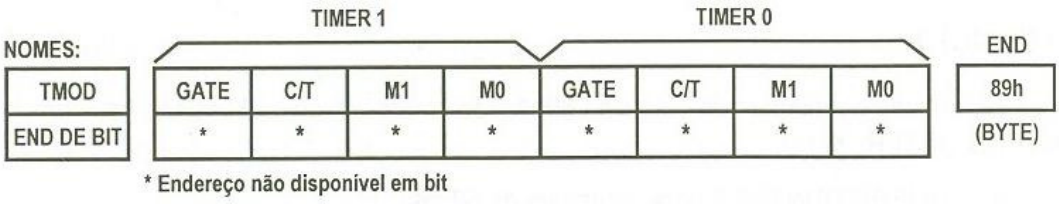


FIGURA 2.9: Registrador TMOD
Fonte: Nicolosi (2004)

- GATE - 1: Refere-se ao disparo de interrupções no *Timer* 1.
- C/T – 1: Em estado 0, utiliza o sinal interno do microcontrolador para contagem no *Timer* 1, em estado 1, utiliza sinal externo.
- M1 – 1 e M0 -1: Define o modo de operação do *Timer* 1.
- GATE - 0: Refere-se ao disparo de interrupções no *Timer* 0.
- C/T – 0: Em estado 0, utiliza o sinal interno do microcontrolador para contagem no *Timer* 0, em estado 1, utiliza sinal externo.
- M1 – 0 e M0 -0: Define o modo de operação do *Timer* 0.

Os *Timers* podem operar em quatro modos distintos, o modo 0, modo 1, modo 2, e modo 3, sendo configurados da seguinte maneira.

Tabela 2.3: Configuração dos modos do *Timer*

Modo	M1	M0	Descrição
0	0	0	Contador de 13 bits
1	0	1	Contador de 16 bits
2	1	0	Contador de 8 bits com auto-reload
3	1	1	Timer misto

- Modo 0: É uma herança da "antiga linha predecessora do 8051, denominada 8048." (Nicolosi, 2004, p. 160). É um contador com capacidade máxima de 13 *bits*.
- Modo 1: Contador de 16 *bits* utilizando o TH e TL respectivos de cada *Timer*.
- Modo 2: Contador de 8 *bits* com capacidade de *auto-reload*, ou seja, ele utiliza um dos registradores que compõe o *Timer* para a contagem de estouro, permitindo assim que a contagem não seja interrompida.

- Modo 3: Permite a contagem de eventos em 8 *bits* e temporizador de 8 *bits*.

3. DESENVOLVIMENTO DO PROJETO

Neste capítulo é abordada tanto a parte física quanto a parte lógica do projeto, pois é com a junção de ambas que foi possível o desenvolvimento do protótipo do ciclocomputador.

Na parte física é tratado o microcontrolador utilizado, o *display* e a relação entre eles e os outros componentes do circuito, permitindo assim o entendimento da placa e do *datasheet* do ciclocomputador.

Já na parte lógica, é apresentada a linguagem utilizada, o montador e o gravador, além de explicar cada função criada na programação, permitindo a compreensão dos fluxogramas e do algoritmo.

3.1. PARTE FÍSICA

Aqui serão abordados o microcontrolador utilizado no protótipo, o AT89C2051 e o *display LCD*. Após esta primeira parte, será abordado o hardware do ciclocomputador, onde será explicado o relacionamento existente entre o microcontrolador, o *display* e os demais componentes.

3.1.1. MICROCONTROLADOR AT89C2051

O AT89C2051 é um microcontrolador de alta performance da família 8051 de 20 pinos desenvolvido pela ATMEL. De acordo com ATMEL (2005), o AT89C2051 possui 2

Kbytes de memória *Flash*, 128 *bytes* de memória RAM; dentre seus 20 pinos, 15 são para entrada e saída, dois são *timer/counters* de 16 bits, um é oscilador interno, porta serial *full-duplex*, opera com alimentação de 2.7V a 6V e com freqüência até 24MHz e a corrente máxima de trabalho por pinos é de 20mA, sendo que para todos os pinos é de 80mA.

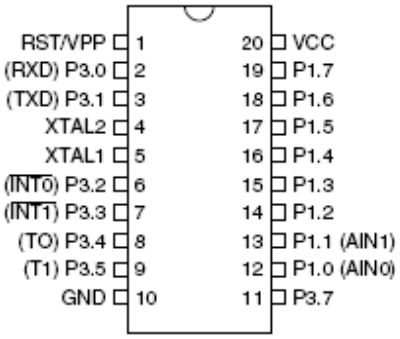


FIGURA 3.1: Organização dos pinos do 2051
Fonte: ATMEL (2005)

Seus pinos possuem as seguintes descrições:

- VCC: Pino que recebe a alimentação.
- GND: Pino ligado ao aterramento.
- *Port* 1: Compreende os pinos P1.0 ao P1.7. É uma *port* utilizada para entrada e saída bidirecional de 8 *bits*. No protótipo, eles formam, junto aos pinos 7 a 14 do *display*, o barramento de dados e instrução.
- Port 3: Compreende os pinos P3.0 ao P3.5, mais o P3.7, e também são utilizados para entrada e saída. O pino P3.6 não é disponível para entrada e saída, pois é utilizado pelo comparador analógico do microcontrolador. No ciclocomputador, o P3.0 envia para o

d*isplay* o sinal de dado ou instrução, o P3.1 habilita ou desabilita do *display*, P3.2 e P3.3 não são utilizados no protótipo, P3.4 e P3.5 são ligados aos botões e o P3.7 é ligado ao sensor.

- RST: Pino que recebe o sinal de *reset* do circuito.
- XTAL1: Entrada do amplificador inverso do oscilador.
- XTAL2: Saída do amplificador inverso do oscilador.

O AT89C2051 possui 128 *bytes* de memória RAM endereçável, fora a área de registradores especiais. Foi utilizado dela o banco 0 da área de banco de registradores, a posição 21 a 26 para o armazenamento da distância, 27 a 29 para a velocidade, 2A a 2C para a massa, 2D a 30 para a caloria e 33 a 43 para as demais necessidades do programa.

Foram usados o *Timer* 0 e *Timer* 1, ambos trabalhando no modo 1, utilizando o *clock* interno do microcontrolador.

A ligação do microcontrolador com os demais circuitos presentes no protótipo será descrita adiante.

3.1.2. *DISPLAY LCD*

O *display LCD* é usado como interface de saída de um sistema. Podem ser de gráficos ou de caracteres. O utilizado no protótipo é um *display* de duas linhas por dezesseis caracteres, que possui 14 pinos com as seguintes funcionalidades:

Tabela 3.1: Pinos do *display LCD*

Pino	Função	Descrição
1	Alimentação	Terra ou GND
2	Alimentação	VCC ou +5V
3	V0	Tensão de ajuste de contraste
4	RS Seleção	1 - Dado, 0 - Instrução
5	R/W Seleção	1 - Leitura, 0 - Escrita
6	E Chip Select	1 - Habilita, 0 - Desabilita
7	B0 LSB	Barramento de dados
8	B1	
9	B2	
10	B3	
11	B4	
12	B5	
13	B6	
14	B7 MSB	

Fonte: Barbacena; Fleury (1996)

Cada posição dos caracteres no *display* também possui seus endereços, assim é possível indicar onde se quer que um caractere seja escrito.

Tabela 3.2: Endereços dos caracteres

LCD 16 x 2	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Linha 1	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
Linha 2	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF

Fonte: Barbacena; Fleury (1996)

O *display LCD* possui uma lista de instruções em hexadecimal para seu funcionamento que, no caso do protótipo, foram programadas junto com as rotinas do *software* e durante o funcionamento são passadas pelo microcontrolador ao LCD. No programa foram utilizadas as seguintes instruções:

Tabela 3.3: Instruções utilizadas

Descrição	Modo	Código da Instrução (h)
Limpeza do display com retorno do cursor		01
Controle do cursor	Inativo	0C
Sentido de deslocamento do cursor	Direita	06
Endereços da primeira posição do display	Linha 1	80
	Linha 2	C0

3.1.3. *HARDWARE*

Depois de descrito o AT89C2051 e o *display LCD*, é possível explicar o relacionamento de ambos com os demais componentes encontrados no *hardware* do ciclocomputador, como são mostrados no *datasheet* do ciclocomputador, no apêndice A.

3.1.3.1. BOTÕES

Foram usados dois botões, ambos com a finalidade de inserir a massa. Um funciona como "*enter*", com a função de habilitar a inserção da massa, passar da centena para dezena e da dezena para unidade, depois sair da inserção e um botão de incremento, que incrementa o valor de cada posição da massa. O botão "*enter*" está ligado ao pino P3.4, enquanto o de incremento é ligado ao P3.5.

Os dois pinos trabalham com uma corrente máxima de 20mA. Para assegurar que a corrente que entra nos pinos não seja maior que a suportada, é utilizado um resistor para *pull-up* externo. Além de assegurar a corrente, o resistor de *pull-up* permite que ela não flutue aleatoriamente, garantindo assim um nível lógico para o pino.

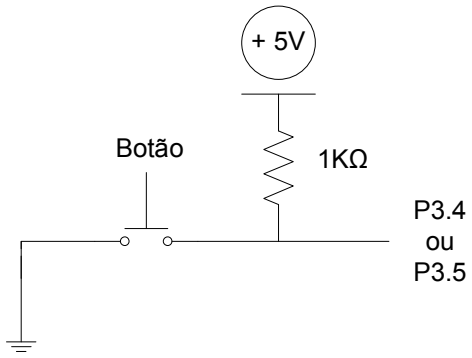


FIGURA 3.2: *Pull-up* do botão

No circuito, é usado um resistor de 1KΩ. Com ele assegura-se uma corrente de 5mA chegando ao pino, deixando-o em nível lógico 1, ou seja, com passagem de corrente. Quando o botão é acionado, o nível do pino cai para 0, pois a corrente deixa de ir para o pino e vai em direção ao aterramento, permitindo que seu acionamento seja reconhecido pelo microcontrolador.

3.1.3.2. SENSOR

É usado um sensor magnético *reed-switch* para a captura dos movimentos da roda da bicicleta. O sensor funciona como uma chave que está aberta. Quando ocorre a passagem do imã próximo ao sensor, a chave se fecha, fechando assim o circuito.

FIGURA 3.3: Sensor *reed-switch* e imã

Ele é posicionado na forquilha superior da bicicleta, próximo aos aros da roda traseira. Foi escolhida a roda traseira para se colocar o sensor devido à maior praticidade para a exibição à banca examinadora.

São usados dois imãs, cada um fixado em um aro da roda traseira, em um ângulo de 180° em relação ao outro e na altura de onde o sensor foi colocado na forquilha. A utilização de dois imãs ao invés de apenas um permite que o sensor conte meia volta da roda, ao invés de uma volta completa.

O sensor é ligado ao microcontrolador utilizando o pino P3.7, e utiliza o mesmo sistema de *pull-up* descrito para os botões, usando também um resistor de 1KΩ.

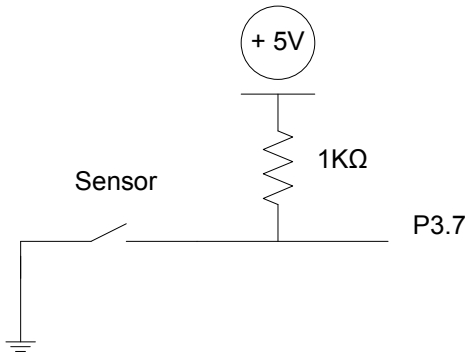


FIGURA 3.4: *Pull-up* do sensor

3.1.3.3. *CLOCK*

Segundo Nicolosi (2004), o microcontrolador possui um circuito oscilador, que depende externamente de uma ligação com um cristal e com capacitores, formando assim o *clock*. É usado um cristal de 12MHz pois com ele o ciclo de máquina do microcontrolador fica em 1μs, permitindo uma contagem precisa do tempo. Ainda de acordo com Nicolosi (2004), os dois capacitores necessários para o funcionamento do *clock* são de 30pF, podendo variar em 10pF para mais ou para menos. Os utilizados são de 33pF. A ligação do cristal e dos capacitores é feita pelos pinos XTAL1 e XTAL2 do microcontrolador.

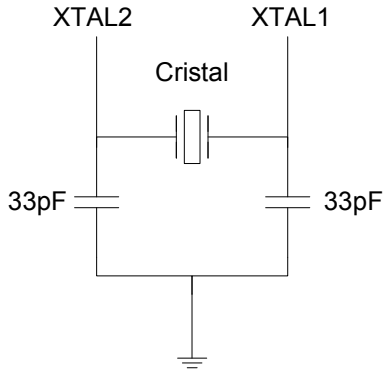


FIGURA 3.5: Circuito oscilador

3.1.3.4. *RESET*

O *reset* utilizado no protótipo é o *reset* automático, o que acontece quando o *chip* é energizado. Para que o *reset* ocorra, é necessário que o pino RST do microcontrolador receba o nível lógico 1 durante pelo menos 2 ciclos de máquina. Quando ocorre, todos os pinos do microcontrolador assumem nível lógico 1. De acordo com Nicolosi (2004), para o circuito, é preciso um capacitor de 10μF e um resistor de pelo menos 8KΩ para gerar o sinal por 2 ciclos de máquina. No circuito do ciclocomputador, foi utilizado o capacitor de 10μF um resistor de 10KΩ.

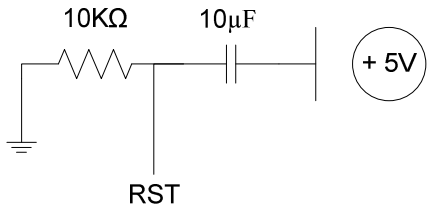


FIGURA 3.6: Circuito de *Reset*

3.1.3.5. LIGAÇÕES DO *DISPLAY LCD*

O *display* interage com o microcontrolador e com o controle de contraste. Seguindo os pinos do display, temos:

- Pino 1: Ligação com o aterramento;
- Pino 2: Ligação com a alimentação;
- Pino 3: Ligação com o controle de contraste. O controle de contraste serve para melhorar a visibilidade do *display*, permitindo o controle da passagem de tensão no pino. O circuito do controle de contraste é o seguinte:

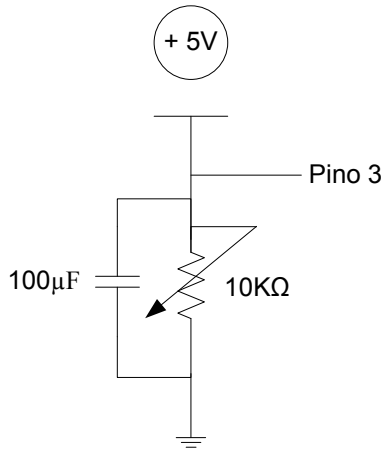


FIGURA 3.7: Controle de contraste

- Pino 4: É responsável por indicar ao *display* se a informação que ele recebe no momento é um dado ou uma instrução. Para tanto, quando ele recebe nível lógico 1, ele informa que é dado e quando recebe 0, informa que é instrução. Quem envia os sinais ao pino 4 do *display* é o pino P3.0 do microcontrolador.

- Pino 5: É responsável por informar ao *display* se é para realizar a escrita dos dados ou para realizar a leitura. No protótipo, o *display* é utilizado apenas para a exibição de informações, portanto o pino é ligado diretamente ao aterramento, fazendo com que ele receba sempre o nível lógico 0, que é o nível para escrita.

- Pino 6: É responsável por habilitar ou não o *display*. Em nível lógico 0, o *display* é desabilitado e em nível lógico 1, habilitado. Quem envia os sinais ao pino 6 é o pino P3.1 do microcontrolador.

- Pinos 7 a 14: São pinos que recebem as informações para o *display*. São ligados aos pinos P1.0 a P1.7, formando assim o barramento de dados e instruções do ciclocomputador.

3.1.3.6. ALIMENTAÇÃO

O protótipo precisa de uma alimentação de 5V para seu funcionamento. Como é utilizada uma bateria de 9V, foi necessária a inclusão de um regulador de tensão. Ele permite que, não importando a variação da tensão de entrada, sua saída sempre seja de 5V contínua.

A seguir é apresentado imagens do protótipo, uma durante o desenvolvimento, utilizando o *protoboard*, e a segunda com a placa já feita.

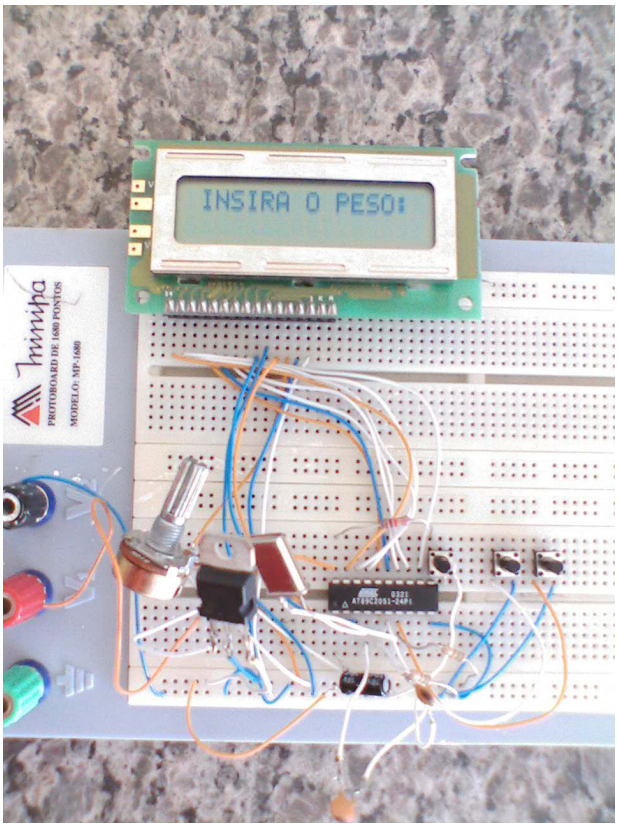


FIGURA 3.8: *Hardware* do protótipo trabalhado no *protoboard*

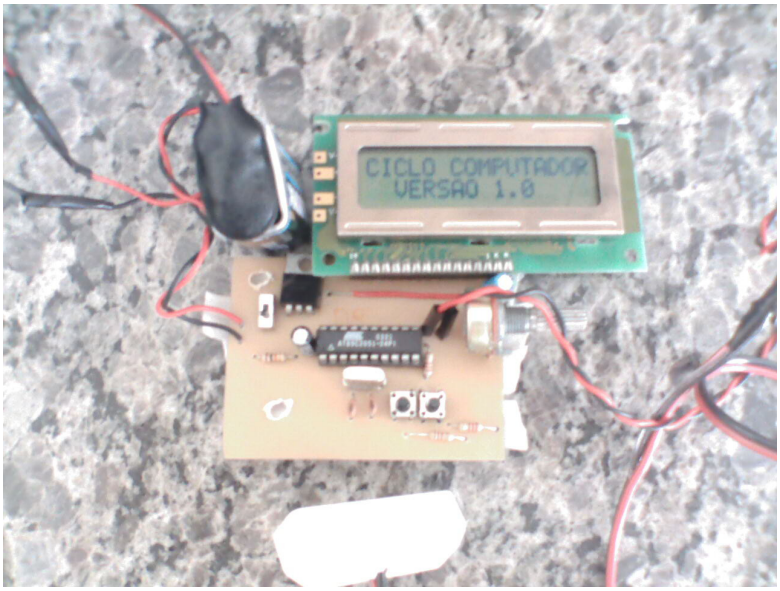


FIGURA 3.9: *Hardware* do protótipo finalizado

3.2. PARTE LÓGICA

Nesta seção serão abordadas as etapas referentes à programação do protótipo, com uma visão rápida da linguagem utilizada, do funcionamento do montador e do gravador e a descrição de cada função do algoritmo, obtendo assim uma melhor compreensão do mesmo.

3.2.1. *ASSEMBLY*

Microcontroladores utiliza a linguagem de máquina, onde “as instruções e dados são trabalhados em nível binário.” (Nicolosi, 2000, p. 195). É um nível muito baixo de programação, de difícil compreensão ao ser humano.

Neste trabalho foi adotada a linguagem *Assembly* para a programação. Ela é uma linguagem estruturada um nível acima da linguagem de máquina, sendo considerada uma linguagem de baixo nível. Nesta seção é abordada a estrutura da programação e os códigos utilizados no algoritmo, possibilitando assim uma melhor compreensão do mesmo.

O *Assembly* trabalha diretamente com os registradores e locais de memória, o que reduz o nível de abstração do programa, mas é eficiente, tornando-a a linguagem preferencial em ambientes onde a velocidade de execução e o tamanho do programa executável são críticos.

Ela possui estrutura e conjunto de instruções idênticas as da linguagem de máquina, porém facilita a programação por poder utilizar os mnemônicos, nomes dados aos *opcodes* (código binário ou hexadecimal das operações) das instruções, e símbolos no lugar de pura lógica binária.

De acordo com Ricarte (2000), o formato geral de uma linha de comando em *Assembly* é:

[rótulo] operação [operando[,operando,...]] [comentário]

Os colchetes indicam campos que não são obrigatórios em todas as instruções.

Durante a programação o rótulo é colocado como referência para que *loops* sejam executados. A operação é a instrução que será executada, os operandos são os parâmetros utilizados para a execução da instrução, podendo variar de valores: decimais, binários, hexadecimais, locais de memória, registradores, portas, ponteiros dentre outros. Cada operação

indica os operandos necessários para sua execução. Os comentários são utilizados para descrever o que a linha do programa executa, permitindo que ela seja mais legível.

Podem-se usar números de bases diferentes como constantes. Portanto cada uma possui uma maneira de ser escrita, permitindo assim a correta interpretação da operação. Por exemplo, a passagem do número 10 como operando pode ser feita das seguintes formas:

- Decimal: #10
- Binário: #%1010
- Hexadecimal: #$0A ou #0Ah

A seguir estão as tabelas contendo os mnemônicos utilizados na programação do protótipo e sua codificação utilizada nas mesmas.

- Rn: Indica registro R0 a R7 genericamente.
- Ri: Indica registro R0 ou R1.
- @: Ponteiro.
- #DADO: Indica valor constante de 8 bits.
- #DADO 16: Indica valor constante de 16 bits.
- DIRETO: Indica um endereço de memória 8 bits.
- rel: Indica que o endereçamento é relativo.

Tabela 3.4: Mnemônicos de Operações Aritméticas

Mnemônicos	Função
ADD A, DIRETO	Soma o conteúdo da posição de memória ao acumulador
ADD A, #DADO	Soma o dado ao acumulador
SUBB a, #DADO	Subtrai o dado e o Borrow do acumulador
INC A	Soma 1 ao acumulador
INC Rn	Soma 1 ao conteúdo de Rn
INC DIRETO	Soma 1 à posição de memória
INC @Ri	Soma 1 à RAM endereçada por Ri
DEC Rn	Subtrai 1 do conteúdo de Rn
DEC DIRETO	Subtrai 1 do conteúdo da posição da memória
MUL AB	Multiplica A e B. Resultado: BA
DIV AB	Divide A e B. Resultado: A inteiro e B resto

Fonte: Júnior. Práticas do Microcontrolador 8051 (1998)

Tabela 3.5: Mnemônicos de Transferência de Dados

Mnemônicos	Função
MOV A, Rn	Mova o registro para o acumulador
MOV A, @Ri	Mova RAM endereçada por Ri ao acumulador
MOV A, #DADO	Mova o dado para o acumulador
MOV Rn, A	Mova o acumulador para o registro
MOV Rn, #DADO	Mova o dado para o registro
MOV DIRETO, A	Mova acumulador para a memória
MOV DIRETO, Rn	Mova o registro para a memória
MOV DIRETO 1, DIRETO 2	Mova o conteúdo da memória direto 2 para a memória direta 1
MOV DIRETO, #DADO	Mova o dado para a memória
MOV @Ri, #DADO	Mova o dado para a RAM indireta
MOV DPTR, #DADO 16	Mova dado de 16 bits para o DPTR
MOVC A, @A + DPTR	Soma A + DPTR obtendo um endereço de 16 bits na memória de programa. Carrega acumulador com esta memória

Fonte: Júnior. Práticas do Microcontrolador 8051 (1998)

Tabela 3.6: Mnemônicos de Variáveis Booleanas

Mnemônicos	Função
CLR C	Zera o Carry
CLR bit	Zera o bit endereçado
SETB C	Seta o Carry
JC rel	Desvia se o carry estiver setado
JNC rel	Desvia se o Carry estiver zerado
JB bit, rel	Desvio se o bit endereçado estiver setado
JNB bit, rel	Desvia se o bit endereçado estiver zerado

Fonte: Júnior. Práticas do Microcontrolador 8051 (1998)

Tabela 3.7: Mnemônicos de Instrução de Desvio

Mnemônicos	Função
LCALL END 16	Chama sub-rotina em qualquer posição da memória de programa
RET	Retorne da sub-rotina
LJMP END 16	Desvia para qualquer posição da memória de programa
JNZ, rel	Desvia se o acumulador "não for zerado"
CJNE A, DIRETO, rel	Compara e desvia se o acumulador for direfente da memória endereçada
CJNE A, #DADO, rel	Compara e desvia se o acumulador for direfente do dado
CJNE Rn, #DADO, rel	Compara e desvia se o registro for direfente do dado
CJNE @Ri, #DADO, rel	Compara e desvia se a RAM indireta for direfente do dado
DJNZ Rn, rel	Decrementa o registro e desvia se for "diferente" de zero
DJNZ DIRETO, rel	Decrementa da memória e desvia se for "diferente" de zero

Fonte: Júnior. Práticas do Microcontrolador 8051 (1998)

3.2.2. MONTADOR E GRAVADOR

Para programas desenvolvidos em *Assembly*, é utilizado um montador (*Assembler*) para converter o código em linguagem de máquina. O montador utilizado foi o AVMAC51, produzido pela AVOCET System.

Ele é formado por 4 arquivos: AVLIB, AVLINK, AVMAC51, AVREF, sendo que os arquivos utilizados para a montagem são o AVMAC51 e o AVLINK trabalhando em ambiente *prompt*. Ainda é necessário o arquivo HEXABIN, para a conversão em binário para posterior gravação no microcontrolador.

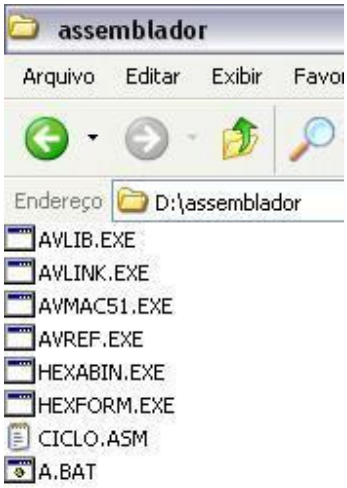


FIGURA 3.10: Estado da pasta inicialmente

O arquivo AVMAC51 é responsável por verificar se há erros na linguagem do algoritmo, além de gerar o arquivo "*.obj", utilizado pelo AVLINK.

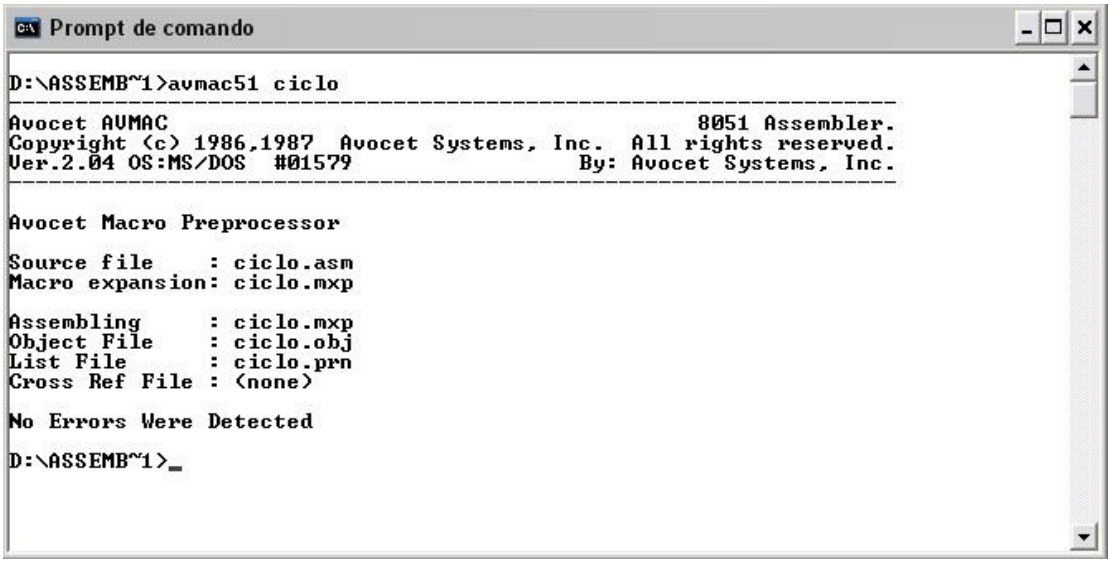


FIGURA 3.11: Execução do arquivo AVMAC51

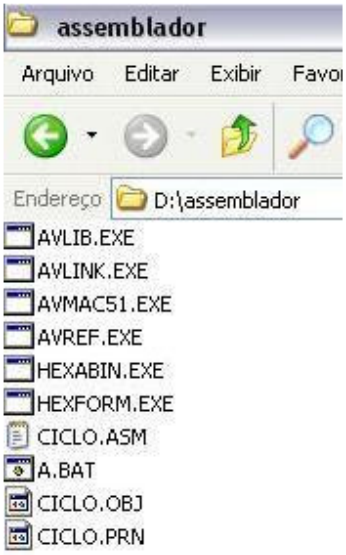


FIGURA 3.12: Estado da pasta após a execução do AVMAC51

Após a execução do AVMAC51, executa-se o AVLINK, que é responsável por "linkar" as sub-rotinas à rotina principal e gerar o arquivo "*.hex", conversão do arquivo original em hexadecimal.

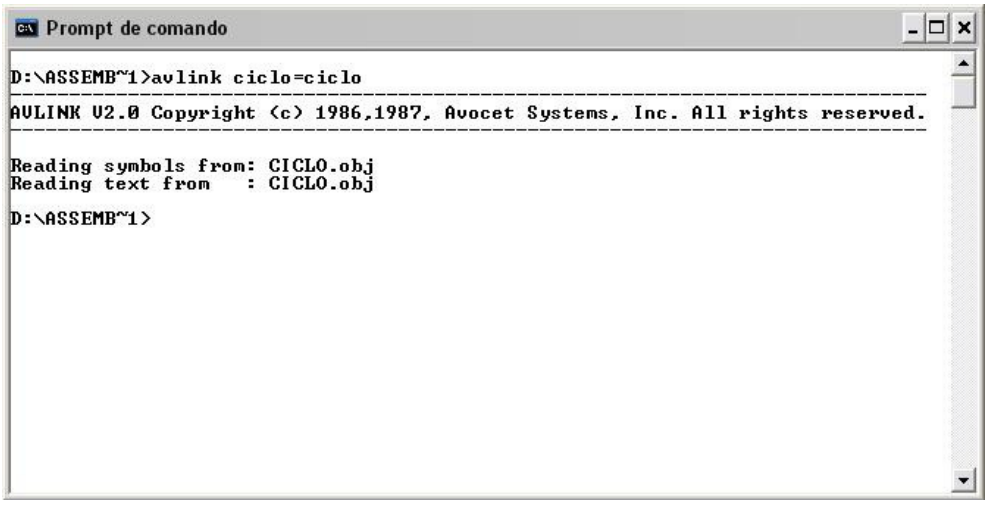


FIGURA 3.13: Execução do arquivo AVLINK

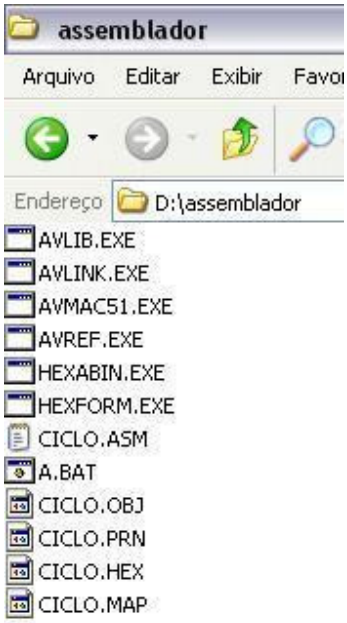


FIGURA 3.14: Estado da pasta após a execução do AVLINK

Finalizada a execução do AVLINK, executa-se o HEXABIN, que utiliza o arquivo "*.hex" para obter o arquivo "*.bin", arquivo este utilizado para a gravação no microcontrolador.

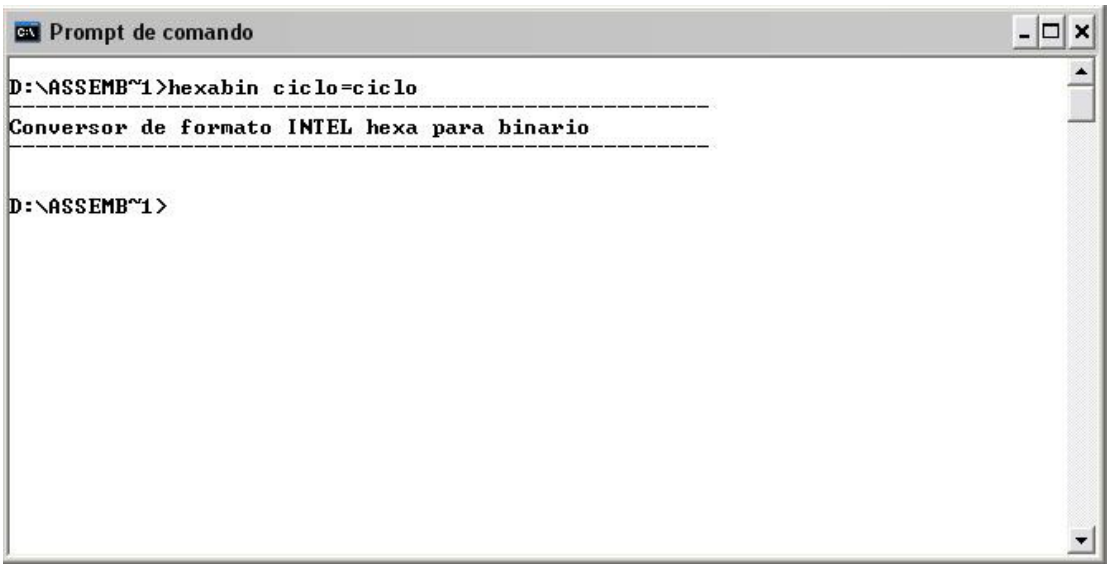


FIGURA 3.15: Execução do arquivo HEXABIN

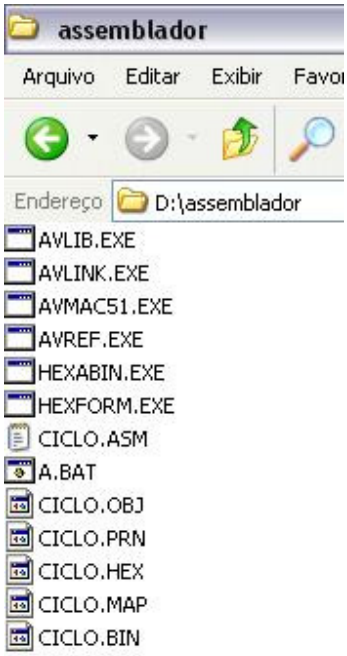


FIGURA 3.16: Estado da pasta após a execução do HEXABIN

Para simplificar o processo, pode-se criar um arquivo "*.bat" contendo as rotinas do AVMAC51, AVLINK e HEXABIN, executando assim apenas um comando e obtendo a verificação de erro, a "linkagem" das rotinas e o arquivo "*.bin".

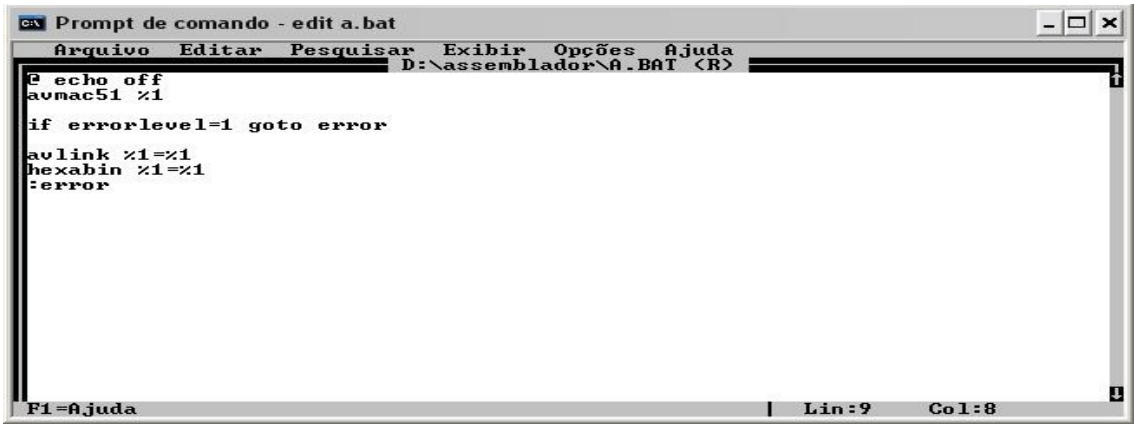


FIGURA 3.17: Edição do arquivo a.bat

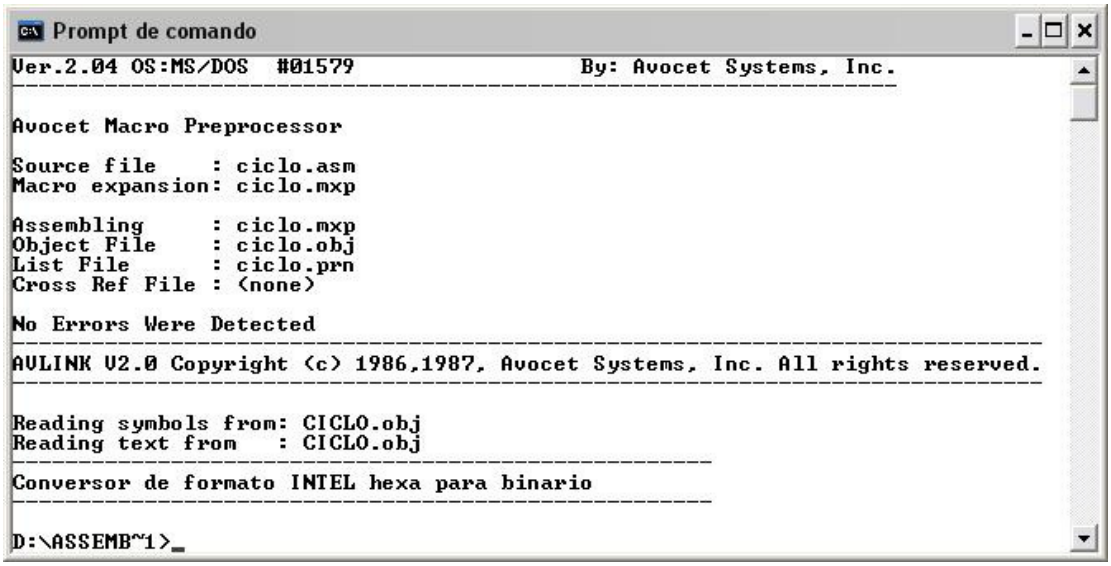


FIGURA 3.18: Execução do arquivo a.bat

Utilizando os três arquivos ou o arquivo editado, no final do processo, obtém-se o programa montado e convertido para "*.bin", estando assim pronto para ser gravado.

Foi utilizado o gravador MACSYM para microcontroladores AT89CX051, compatível com o AT89C2051 utilizado no protótipo. O *hardware* do gravador é ligado à porta paralela do computador e o microcontrolador é acoplado a ele, seguindo a ordem dos pinos, e o *software* é acessado via *prompt*, onde é realizada uma verificação de leitura do programa "*.bin" e onde a memória flash do microcontrolador é apagada e gravada com este programa.

FIGURA 3.19: *Hardware* do Gravador

FIGURA 3.20: *Software* do Gravador realizando verificação de leitura

3.2.3. *SOFTWARE*

A idéia inicial parecia simples: o usuário iria inserir sua massa e ela seria guardada em um espaço da memória, o sensor captaria o numero de voltas da roda também e armazenaria

em um espaço de memória. Como a circunferência, o tempo e o MET já estariam programados, bastaria utilizar estes cinco dados para o cálculo da distância, velocidade instantânea e perda calórica. Mas durante o desenvolvimento, verificou-se que não era tão simples assim.

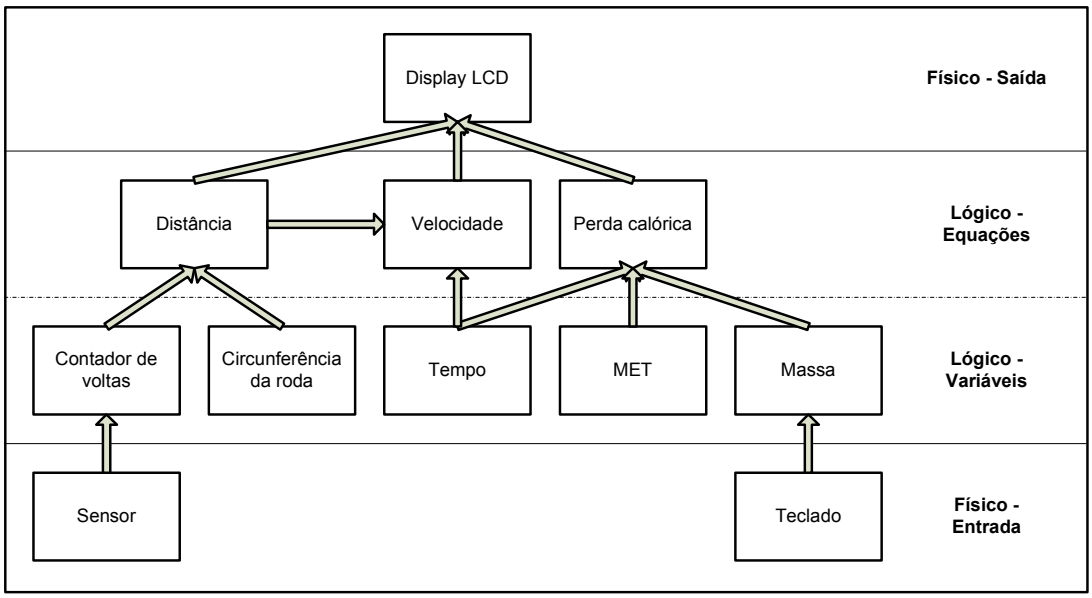


FIGURA 3.21: Datagrama do ciclocomputador

Nesta subseção serão tratadas as rotinas utilizadas no programa. É sugerido que se faça a leitura desta subseção acompanhando paralelamente os anexos B e C, fluxogramas e algoritmo respectivamente.

As rotinas serão divididas em blocos assim como se encontram no algoritmo impresso e nos fluxogramas em anexo deste trabalho. As rotinas pertencentes a cada bloco serão explicadas de maneira geral, possibilitando uma melhor compreensão dos anexos.

É valido salientar a importância de 2 destes blocos, os referentes à velocidade e caloria, por serem os que envolveram as maiores adaptações matemáticas e cálculos binários, sendo assim mais complexos.

3.2.3.1. ROTINAS DE *DELAY*

As rotinas presentes neste bloco são usadas para gerar atrasos, permitindo assim a contagem ou a passagem de tempo necessária.

As rotinas DELAY e TEMPO funcionam da mesma maneira, a diferença são os valores para os quais cada uma é inicializada.

No DELAY, o objetivo é criar uma espera de 1000μs, utilizado na espera do sensor para que seja contabilizado apenas um acionamento quando o ímã o ativa e para o atraso que o *display* necessita para a compreensão de dados e instruções. Como é utilizado o *Timer* em modo 1, que é um contador de 16 *bits*, sua contagem chega a 0 a 65535 (2^{16}); iniciando o *Timer* com o valor hexadecimal FC17h (64535), gera uma contagem de 1000 (65535 – 64535 = 1000) e com o ciclo de máquina de 1μs, a espera desejada de 1000μs é conseguida.

No TEMPO é gerada uma espera de 62500μs, usada para criar um atraso para a rotina MENS_INICIAL, afim de ela aparecer por alguns instantes no *display*. O *Timer* é iniciado com 0BDBh (3035), o que gera uma contagem de 62500.

As duas rotinas utilizam o Timer0 e são ligadas e encerradas quando são chamadas.

A TEMP também gera uma espera de 62500µs para a contagem de segundos, basta chamar a rotina por dezesseis vezes (62500µs * 16 = 1s). A diferença dela com o TEMPO é que ela não é ligada e encerrada quando chamada. Esses dois parâmetros são determinados na rotina principal em que a mesma foi chamada.

3.2.3.2. ROTINAS DE *DISPLAY*

São rotinas padrões para o funcionamento do *display*.

O DADO é utilizado para informar ao *display* que a informação presente no barramento no momento é um dado, enquanto a INSTRUCAO informa que a informação recebida é uma instrução. O *display* precisa de 1000µs para assimilar dados e 3000µs para assimilar instrução. Para gerar esse tempo, nas duas rotinas é chamada a rotina DELAY. O INICIADISP realiza as configurações necessárias para a inicialização do display.

As instruções de *display* utilizadas são encontradas na subseção *Display LCD* deste capítulo, na página 31.

3.2.3.3. ROTINAS DE MENSAGEM

Estas rotinas são usadas para a apresentação de mensagens no *display*.

A MENSAGEM realiza a varredura da *string* de mensagem, retornando, a cada *loop*, um caractere. A MENS_INICIAL realiza a exibição no display da primeira mensagem. A ZERA_DISP joga zeros em um determinado número de posições do display definido pelo seu

valor de entrada. MENS_UNIDADE exibe as mensagens das unidades da velocidade, distância e caloria, além de determinar o valor de entrada do ZERA_DISP. MENS_MASSA exibe a mensagem de inserção da massa do usuário. A EXIBIR_DISP é uma rotina utilizada por outras para a exibição de dados no display.

3.2.3.4. ROTINAS DE INSERIR MASSA

São as rotinas utilizadas para inserir e armazenar o valor da massa do usuário.

A INSE_MASSA é usada para a inserção da massa. Ela exibe zero na posição e espera o acionamento das teclas. Caso seja pressionado o "*enter*", ela sai, caso seja pressionado o incremento, o valor sobe em uma unidade até que atinja seu valor máximo, depois volta a ser zerado. A C_MAS, D_MAS e U_MAS passam os parâmetros necessários para a INSE_MASSA, além de salvarem cada qual seu valor da massa, ou seja, a C_MAS armazena a centena, a D_MAS armazena a dezena e a U_MAS armazena a unidade da massa..

3.2.3.5. ROTINAS DE DISTÂNCIA

A DIST_PERCORRIDA recebe como parâmetro inicial a posição da unidade da distância. Assim toda vez que é chamada, ela incrementa o valor da unidade. Quando o valor chega a 10, zera-o e passa-se para a posição da dezena, e a incrementa. Caso o valor da dezena chegue a 10, o mesmo processo é realizado, zerando-se o valor da posição da dezena e incrementando o valor da centena. Esse processo vai até o valor 999999, valor máximo para a exibição da distância. No próximo incremento, a distância é zerada.

Como dito na subseção Aspectos Matemáticos, foi utilizada como referência para o cálculo da distância uma bicicleta de aro 26", que possui uma circunferência de 2,05 metros. Devido à dificuldade de trabalhar com números decimais, a circunferência foi arredondada arbitrariamente para 2 metros, o que gera um erro de 2,5% a menos na distância. Se for considerada a massa do usuário em cima da bicicleta, ela gera uma deformação no pneu, um achatamento, o que reduz o raio naquele ponto, reduzindo assim a circunferência, o que faz com que o erro de 2,5% seja reduzido ainda mais.

3.2.3.6. ROTINAS DE VELOCIDADE

Aqui estão as rotinas que convertem e exibem a velocidade.

A CONV_VEL realiza a conversão da velocidade de metros por segundo para quilômetros por hora. Ela recebe como entrada a distância percorrida em 1 segundo de exercício. Para a conversão, bastaria realizar a multiplicação da velocidade em m/s pelo fator de conversão "3,6", mas devido às restrições de *hardware* do microcontrolador, foi necessário um maior esforço matemático. O microcontrolador não possui a unidade de ponto flutuante, o que impossibilita a realização dos cálculos com números decimais diretamente. Poderia ter sido feito multiplicando primeiro por 36 e depois dividindo por 10, mas desta maneira a velocidade ficaria restrita ao máximo de 25km/h, pois durante uma multiplicação, é utilizado o acumulador A e o B, e ambos armazenam entre 0 e 255. Como a parte menos significativa da multiplicação fica no A, ele poderia no máximo armazenar o valor 255, o que limitaria a conversão para uma velocidade máxima de 7m/s (7 * 36 = 252), o que resultaria numa exibição máxima de 25 km/h. Por isso a lógica implementada no protótipo foi mais complexa.

A conversão segue a seguinte lógica, permitindo a exibição de uma velocidade máxima de 151 km/h (42m/s), velocidade suficiente para utilização em bicicletas.

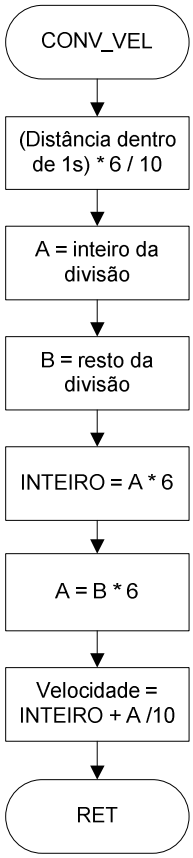


FIGURA 3.22: Fluxograma da CONV_VEL

Por exemplo, se pegarmos uma velocidade de 6m/s e seguirmos os passos, pela divisão obtêm-se inteiro 3 e resto 6, valores atribuídos a A e B respectivamente. INTEIRO recebe 3 * 6 = 18 e o A recebe 6 * 6 = 36. Portanto a Velocidade é 18 + 36 / 10 = 21 km/h, sendo este valor truncado. Se a conversão for feita utilizando o fator 3,6, o valor obtido também é 21 km/h, também considerando o truncamento.

No final desta conversão, o resultado fica armazenado em apenas uma posição de memória, mas para que seja feita sua exibição em três caracteres do display, é necessário que se faça uma conversão para BCD (*Binary Coded Decimal*), ou decimal representado em binário. Essa é a função da rotina DEC_BCD_3D_VEL. Continuando o exemplo anterior, a velocidade 21 km/h ficaria da seguinte maneira:

Tabela 3.8: Conversão BCD

0	2	1	Número Decimal
xxxx 0000	xxxx 0010	xxxx 0001	BCD
a+2	a+1	a	Posição na memória

Por fim, a EXIBIR_VEL passa os parâmetros para essas duas rotinas e as chama, além de passar os parâmetros e chamar a EXIBIR_DISP para exibir a velocidade no display.

3.2.3.7. ROTINAS DE CALORIA

Neste bloco estão todas as rotinas necessárias para o cálculo e exibição da caloria.

Durante a inserção da massa pelo usuário, os valores de unidade, dezena e centena foram colocados um em cada espaço de memória. A CONCAT_MASSA é responsável por juntar essas três informações neste mesmo espaço.

Na proposta apresentada para projeto final, havia sido dito que seria utilizado o valor MET para velocidade abaixo de 16 km/h, ou seja, o valor MET 4. Durante o desenvolvimento, percebeu-se que o usuário poderia alcançar uma velocidade maior que 16 km/h com facilidade, o

que motivou a utilização do valor MET 8, que de acordo com a tabela 2.1 na página 14, é o valor geral para atividade física “andar de bicicleta”.

A fórmula utilizada para o cálculo da caloria é:

Gasto Energético (kcal) = MET x massa (kg) x tempo (min) / 60

Como no ciclocomputador o tempo é contado em segundos e não em minutos, foi necessária uma primeira adaptação:

Gasto Energético (kcal) = MET x massa (kg) x tempo (s)/ 3600

Mais adaptações tiveram que ser feitas na implementação desta fórmula, pois além da falta da unidade de ponto flutuante, existe o problema do estouro da posição de memória. Primeiro a simplificação do valor MET 8 por 3600.

Gasto Energético (kcal) = massa (kg) x tempo (s) / 450

Para uma segunda simplificação, foi necessário fixar o tempo de exibição da caloria. Foi escolhido o tempo de atualização de 30 segundos. Se fosse escolhido um tempo menor, a exibição dos decimais se tornaria mais interessante, mas como há previsão da caloria alcançar o milhar (uma pessoa de 80 kg, segundo a fórmula e o valor MET utilizado, alcança 1000kcal perdidas em 1hora e 30 minutos, aproximadamente), isso se tornou inviável, principalmente pela falta de espaço no display. Além do mais, o tempo de 30 segundos se mostrou satisfatório para a

exibição de calorias, pois o usuário, visando à perda de massa, passa um longo tempo em exercício, tornando uma exibição em tempo menor desinteressante.

Portanto a fórmula acaba ficando da seguinte maneira:

Gasto Energético (kcal) = massa (Kg) / 15

A rotina CONVERT_CAL é chamada apenas se for a primeira exibição da caloria. Ela realiza a divisão por quinze para os primeiros 30 segundos e armazenar o inteiro e o resto desta divisão em espaços de memória diferentes, além de arredondar o valor da caloria caso seja maior que 5.

Como no final da CONVERT_CAL, o gasto energético fica armazenado em apenas um espaço de memória, novas conversões para BCD são necessárias para a exibição no display. A BCD_MENOS_SIG converte a unidade e a dezena.

A INT_CAL é responsável por multiplicar a parte inteira armazenada na CONVERT_CAL pela quantidade de 30 segundos contada e salvar a parte mais significativa do resultado na variável MAIS_SIG e a menos significativa na variável MENOS_SIG.

A REST_CAL realiza a mesma operação da INT_CAL, só que com o resto armazenado na CONVERT_CAL. A parte mais significativa do resultado é armazenada na variável CAL_MAIS_SIG e a menos significativa na variável CAL_MENOS_SIG.

A DIVISAO_P_10 divide o CAL_MENOS_SIG por 10, armazenando a parte mais significativa na variável CAL_MAIS_SIG e a menos significativa na variável

CAL_MENOS_SIG. Se no CAL_MENOS_SIG estivesse armazenado o valor 120, depois de executada, seriam obtidos os valores 12, no CAL_MAIS_SIG, e 0, no CAL_MENOS_SIG.

A CAL_F_TEMPO chama a INT_CAL, a REST_CAL e a DIVISAO_P_10. Depois utiliza a CAL_MENOS_SIG para verificar se a caloria precisa ser arredondada. Caso o arredondamento seja necessário, soma-se 1 a CAL_MAIS_SIG. Então se soma o valor em MENOS_SIG (da INT_CAL) com a CAL_MAIS_SIG, armazenando este valor no MENOS_SIG. Caso haja estouro do *carry* devido à soma, o valor no MAIS_SIG é incrementado.

Simplificando: a CAL_F_TEMPO verifica se há necessidade de arredondamento no valor da caloria e armazena sua parte mais significativa no MAIS_SIG e a menos significativa no MENOS_SIG.

Como agora a caloria está em 16 bits, é necessário realizar uma adaptação nos valores dos dois espaços de memória, para que ao invés do valor total estar escrito em 16 bits, a unidade e dezena estejam nos 8 bits menos significativos e a centena e o milhar estejam nos 8 bits mais significativos. Quem realiza essa divisão de valores é a rotina DIVISAO_P_100.

Por exemplo, após a rotina CAL_F_TEMPO, foi obtido o valor 396. Este valor fica então armazenado em 16 bits da seguinte forma: 00000001 10001100, sendo que no espaço de memória mais significativo ficam 00000001 (1 em decimal) e no menos significativo ficam 10001100 (140 em decimal). Após a rotina DIVISAO_P_100, no espaço mais significativo ficam 00000011 (3 em decimal) e no espaço menos significativo ficam 01100000 (96 em decimal).

Agora que os valores estão da maneira necessária nos espaços de memória, pode-se realizar a conversão para BCD, a fim de exibir o valor no display. Para valores até 255, é

utilizada a rotina a DEC_BCD_3D_CAL. Para valores superiores, é utilizada a rotina BCD_MAIS_SIG em conjunto com a BCD_MENOS_SIG.

E por último a EXIBIR_CAL, que exibe a caloria. Primeiramente ela verifica se são os primeiros 30 segundos, se forem, ela chama a CONVERT_CAL e utiliza a BCD_MENOS_SIG para a conversão. Caso não sejam, são chamadas as CAL_F_TEMPO e DIVISAO_P_100. É verificado se o espaço de memória mais significativo é zero, caso seja, utiliza-se a DEC_BCD_3D_CAL para a conversão, caso não seja, é utilizado a BCD_MENOS_SIG e a BCS_MAIS_SIG. Nos três momentos são fornecidos os parâmetros necessários para cada uma realizar a exibição utilizando a rotina EXIBIR_DISP.

3.2.3.8. ROTINA DE LIMPAR MEMÓRIA

A MEMO_ZERADA é responsável por zerar todas as posições de memória utilizadas pelo programa.

3.2.3.9. *KERNEL* PRINCIPAL

É o principal bloco do programa, o que faz chamada às demais rotinas já explicadas.

A rotina INICIO começa chamando a INICIADISP, para inicializar o display, MENS_INICIAL, para exibir a primeira mensagem, MENS_MASSA, para exibir a mensagem de inserção de massa e a MEMO_ZERADA, para limpar as posições de memória utilizadas. Deste ponto ele aguarda o acionamento da tecla ENTER. Quando acionada, chamam a C_MAS, para

inserir e receber a centena da massa, a D_MAS, para inserir e receber a dezena da massa e a U_MAS, para inserir e receber a unidade da massa. Após a inserção da massa, as demais variáveis utilizadas são inicializadas e a rotina TEMP é chamada, configurando o *Timer* 1. A partir deste momento, o programa fica dependente do acionamento ou não do sensor.

Caso o sensor não seja acionado (momento "A" no fluxograma), é verificado o estouro da flag do *Timer* 1. Como ele só é inicializado da primeira vez quando o sensor é acionado, a rotina volta a esperar o acionamento do sensor, ou seja, só se avança para o momento "A" caso o sensor tenha sido acionado pela primeira vez.

Quando o sensor é acionado (momento "B" no fluxograma), inicializa o N_ASCIO com o valor 3, ele é a variável que mais tarde será usada para verificar se a bicicleta continua em movimento ou se ela parou. Liga-se o *Timer* 1, aciona-se um atraso, para que não haja perigo do sensor contar duas vezes a mesma passada do imã, incrementa-se a distância percorrida em 1 segundo, chama-se a rotina DIST_PERCORRIDA, para o cálculo da distância percorrida, e a EXIBIR_DISP, para exibi-la. Neste ponto volta-se à espera do acionamento do sensor. Então o momento "B" é responsável por carregar a variável N_ASCIO, utilizada para verificar se a bicicleta continua em movimento, ligar a contagem de tempo, incrementa a distância usada para o cálculo de velocidade e atualiza a distância percorrida.

De volta ao instante que o sensor não é acionado (momento "A"), como o Timer 1 foi acionado (no momento "B"), havendo o estouro de sua flag, reinicializa-se o TEMP e espera a passagem de 1 segundo, para calcular e exibir a velocidade. Caso o segundo não tenha sido completado, volta-se à espera do acionamento do sensor.

Caso tenham se passado 1 segundo, a rotina EXIBIR_VEL é chamada, exibindo a velocidade e a distância em 1 segundo é zerada.

Agora através do N_ASCIO, ocorre a verificação se a bicicleta está parada ou em movimento. Caso se passe 3 segundos sem que o sensor seja ativado novamente, o Timer 1 é desligado e a distância em 1 segundo e a contagem de 30 segundos são reinicializadas. Depois a contagem de 1 segundo também é reinicializada. Caso a bicicleta permaneça em movimento, ela avança diretamente para a reinicialização da contagem de 1 segundo.

Ocorre a verificação da contagem de 30 segundos através do TEMP. Se ainda não tiverem passados os 30 segundos, o programa volta a esperar o acionamento do sensor, mas se já tiverem passado tempo, o multiplicador da caloria é incrementado e a caloria é exibida pela EXIBIR_CAL. A contagem de 30 segundos é reinicializada e o programa volta a esperar se o sensor é acionado ou não.

Assim funciona a programação do ciclocomputador.

4. TESTES E RESULTADOS

A fim de verificar o bom funcionamento do protótipo, foram realizados testes comparativos utilizando como parâmetro de comparação os valores apresentados por um ciclocomputador comercializado.

Características do ciclocomputador utilizado, os procedimentos e resultados obtidos com os testes serão apresentados a seguir.

4.1. CICLOCOMPUTADOR CATEYE VEL08

O ciclocomputador utilizado como parâmetro para comparação foi o CATEYE VEL08. Este modelo foi escolhido por apresentar, dentre seus valores de saída, aquelas exibidas pelo protótipo, ou seja, a distância, a velocidade instantânea e a perda calórica.

O CATEYE VEL08 é um ciclocomputador produzido pela empresa japonesa CATEYE, especializada na produção de ciclocomputadores, luzes e refletores para ciclistas.

Este ciclocomputador exibe para o usuário a velocidade em km/h ou mph, tempo decorrido, distância do percurso, velocidade instantânea, média e máxima, consumo de calorias, distância total e relógio.

FIGURA 4.1: CATEYE VEL08

Ele utiliza um sensor magnético que é fixado próximo à roda da bicicleta e um imã, colocado no aro desta roda, de maneira que os dois estejam alinhados.

4.2. MONTAGEM PARA OS TESTES

Foi utilizada uma bicicleta de aro 26” para os testes, pois a definição da circunferência da roda havia sido feita para tal bicicleta.

Usualmente ciclocomputadores são ligados à roda dianteira da bicicleta, exigindo assim menos fiação e tornando a instalação mais fácil. Para os testes, tanto o protótipo quanto o CATEYE VEL08 foram colocados na roda traseira, permitindo que, com o uso de um apoio na forquilha da correia, fazendo com que a roda traseira se elevasse do solo, bastasse o giro dos pedais com as mãos para esta roda começar a girar, evitando assim a necessidade de montar e pedalar a bicicleta, podendo assim evitar possíveis danos ao protótipo.

FIGURA 4.2: Visão geral da estrutura montada para os testes

Para evitar interferência dos imãs para os sensores de ambos os ciclocomputadores, o sensor do protótipo foi instalado na forquilha superior direita, a 8 cm do eixo e o sensor do CATEYE VEL08 na forquilha superior esquerda, a 20 cm do eixo. Essas distâncias suficientes para evitar interferências. Os três imãs, um do ciclocomputador de teste e os outros dois do protótipo, foram instalados em alturas correspondentes aos seus respectivos sensores.

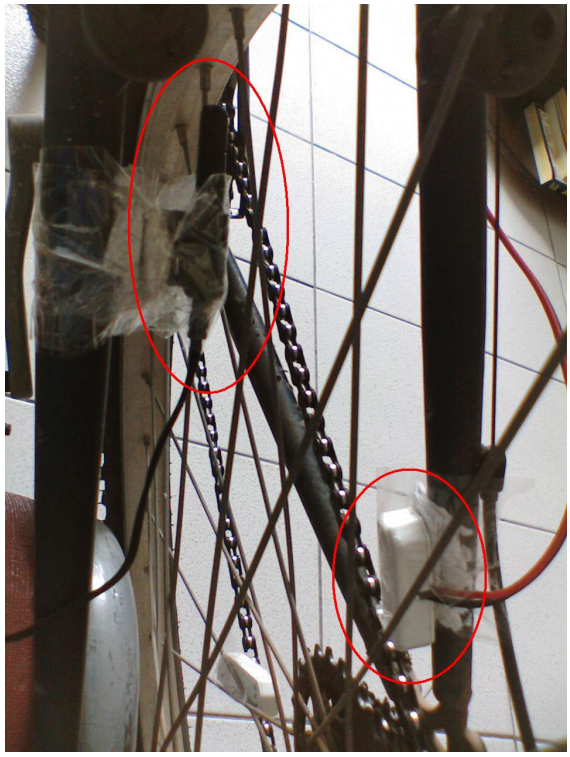


FIGURA 4.3: Sensores instalados.

Para apoio à bicicleta e aos ciclocomputadores foi utilizada uma cadeira.

FIGURA 4.4: Roda e ciclocomputadores acomodados.

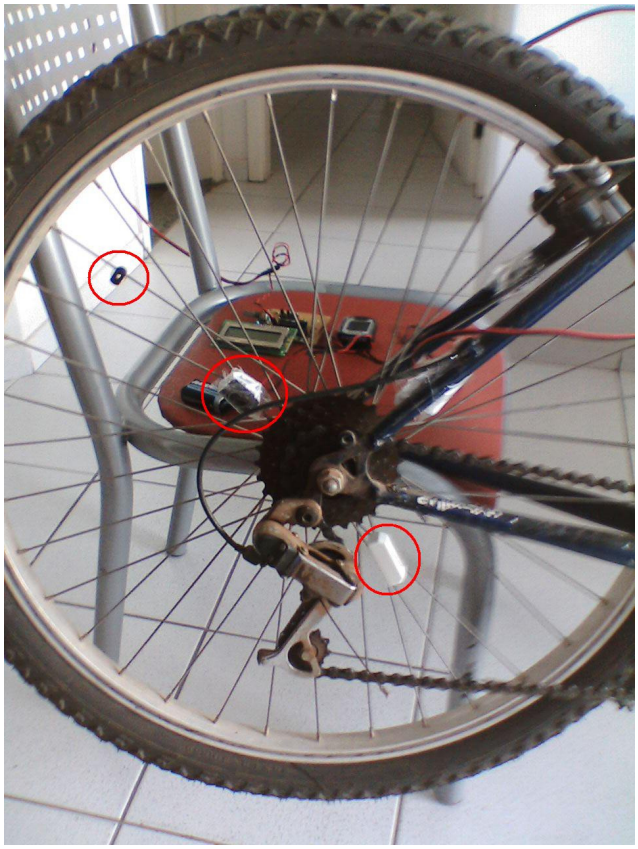


FIGURA 4.5: Imãs fixados

4.3. TESTE DA DISTÂNCIA

Para o teste da distância foram realizadas cinco baterias de testes e em cada uma delas foram feitas tomadas de distância do protótipo a cada 500 metros, chegando a 5000 metros medidos pelo ciclocomputador de testes. Ao atingir o valor esperado no ciclocomputador de testes, a bicicleta era freada e o valor exibido pelo protótipo era anotado. Os resultados obtidos podem ser verificados na tabela a seguir.

Tabela 4.1: Distâncias Tomadas

CATEYE VEL08	Protótipo					Média
	Teste 01	Teste 02	Teste 03	Teste 04	Teste 05	
500	493	491	495	491	494	492,8
1000	984	980	981	981	984	982
1500	1474	1470	1475	1471	1477	1473,4
2000	1964	1964	1965	1963	1967	1964,6
2500	2453	2453	2457	2453	2454	2454
3000	2945	2943	2945	2943	2943	2943,8
3500	3435	3433	3437	3434	3438	3435,4
4000	3925	3930	3931	3920	3927	3926,6
4500	4419	4419	4417	4413	4421	4417,8
5000	4907	4909	4907	4907	4909	4907,8

A variação entre os valores tomados é explicada pela imprecisão ao se frear a bicicleta, além do fato do ciclocomputador de testes apenas exibir a distância percorrida a partir da dezena de metro, impossibilitando saber com maior precisão o momento da freada.

Caso se assumisse a tomada das distâncias no exato momento pretendido, seriam obtidos os seguintes resultados em relação ao seu erro.

Tabela 4.2: Erro Relativo Percentual Considerando as Unidades

CATEYE VEL08	Protótipo					Média	Total
	Teste 01	Teste 02	Teste 03	Teste 04	Teste 05		
500	1,4	1,8	1,0	1,8	1,2	1,4	1,8
1000	1,6	2,0	1,9	1,9	1,6	1,8	
1500	1,7	2,0	1,7	1,9	1,5	1,8	
2000	1,8	1,8	1,8	1,9	1,7	1,8	
2500	1,9	1,9	1,7	1,9	1,8	1,8	
3000	1,8	1,9	1,8	1,9	1,9	1,9	
3500	1,9	1,9	1,8	1,9	1,8	1,8	
4000	1,9	1,8	1,7	2,0	1,8	1,8	
4500	1,8	1,8	1,8	1,9	1,8	1,8	
5000	1,9	1,8	1,9	1,9	1,8	1,8	

O erro médio relativo obtido é de 1,8%. Mas como dito antes, o ciclocomputador de teste exibe a distância a partir de sua dezena, o que gera imprecisão se considerada a unidade. O melhor método para obter um valor de erro mais confiável seria também desconsiderar a unidade dos valores tomados no protótipo, ficando assim com a contagem a partir da dezena de metro.

Tabela 4.3: Distâncias Tomadas Desconsiderando as Unidades

CATEYE VEL08	Protótipo					Média
	Teste 01	Teste 02	Teste 03	Teste 04	Teste 05	
500	490	490	490	490	490	490
1000	980	980	980	980	980	980
1500	1470	1470	1470	1470	1470	1470
2000	1960	1960	1960	1960	1960	1960
2500	2450	2450	2450	2450	2450	2450
3000	2940	2940	2940	2940	2940	2940
3500	3430	3430	3430	3430	3430	3430
4000	3920	3930	3930	3920	3920	3924
4500	4410	4410	4410	4410	4420	4412
5000	4900	4900	4900	4900	4900	4900

Tabela 4.4: Erro Relativo Percentual Desconsiderando as Unidades

CATEYE VEL08	Protótipo					Média	Total
	Teste 01	Teste 02	Teste 03	Teste 04	Teste 05		
500	2,0	2,0	2,0	2,0	2,0	2,0	2,0
1000	2,0	2,0	2,0	2,0	2,0	2,0	
1500	2,0	2,0	2,0	2,0	2,0	2,0	
2000	2,0	2,0	2,0	2,0	2,0	2,0	
2500	2,0	2,0	2,0	2,0	2,0	2,0	
3000	2,0	2,0	2,0	2,0	2,0	2,0	
3500	2,0	2,0	2,0	2,0	2,0	2,0	
4000	2,0	1,8	1,8	2,0	2,0	1,9	
4500	2,0	2,0	2,0	2,0	1,8	2,0	
5000	2,0	2,0	2,0	2,0	2,0	2,0	

Portanto o protótipo possui um erro de menos 2% na exibição da distância. Era esperado um valor de erro máximo de menos 2,5% devido o arredondamento da circunferência. Outro ponto válido a ser lembrado é que o teste foi realizado sem ter a interferência da massa da pessoa, o que achataria o pneu, reduzindo assim o raio da roda e a sua circunferência, que por sua vez influenciaria na obtenção do erro total.

A seguir estão algumas fotos tiradas durante o teste.

FIGURA 4.6: 500 metros, Teste 01

FIGURA 4.7: 1000 metros, Teste 03

FIGURA 4.8: 1500 metros, Teste 04

4.4. TESTE DA VELOCIDADE

Pela análise do teste da distância, tem-se que esse erro da distância só seria sentido na velocidade em valores elevados. A velocidade de 180 km/h, exibida pelo protótipo tem erro de 3,6 km/h, considerando o erro de 2% da distância. Assim, a velocidade de 90 km/h apresenta erro de 1,8 km/h e a velocidade de 45 km/h, erro de 0,9 km/h.

Para velocidades inferiores o erro da distância praticamente não influenciaria no resultado.

Como dito anteriormente, a velocidade é exibida em múltiplos de "3,6", pois a velocidade do protótipo é obtida em metros por segundo, depois multiplicada por esse fator para conversão em quilômetros por hora. Escolheu-se não arredondar o valor da velocidade, mantendo assim apenas seu valor inteiro. O protótipo a exibe da seguinte maneira:

Tabela 4.5: Exibição da Velocidade no Protótipo

m/s	Km/h
1	3
2	7
3	10
4	14
5	18
6	21
7	25
8	28
9	32
10	36

Com o teste confirmou-se que o protótipo exibe a velocidade corretamente dentro de seus parâmetros de exibição, pois nessas velocidades instantâneas o ciclocomputador de teste

também exibia esses valores, mostrando assim que o arredondamento da velocidade não era necessário.

A seguir estão algumas fotos tiradas durante o teste para a confirmação da exibição.

FIGURA 4.9: Velocidade 7km/h

FIGURA 4.10: Velocidade 14km/h

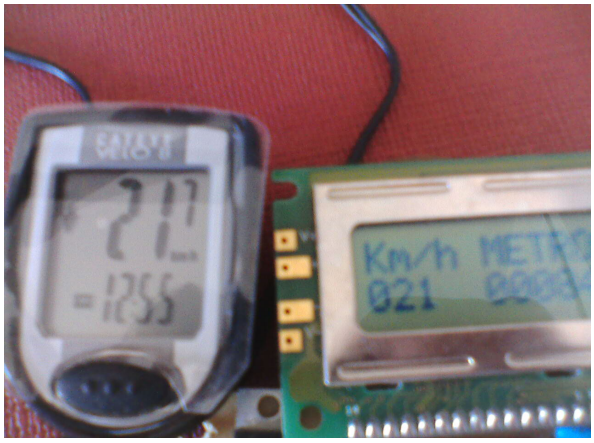


FIGURA 4.11: Velocidade 21km/h

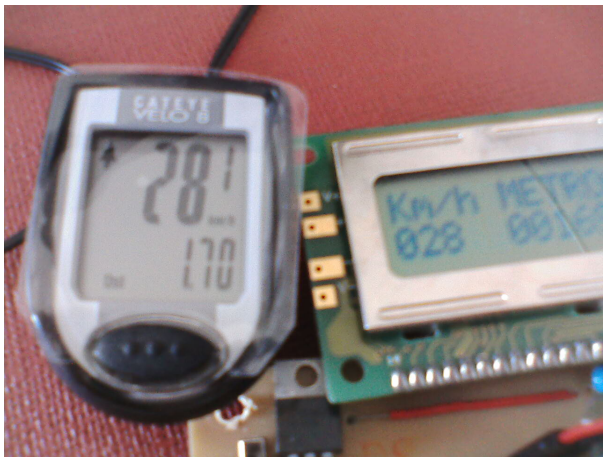


FIGURA 4.12: Velocidade 28km/h

4.5. TESTE DA PERDA CALÓRICA

De acordo com o manual do ciclocomputador CATEYE VEL08, o consumo de calorias é calculado a partir dos dados da velocidade. Não é definida a equação utilizada para este cálculo. Além do mais, este ciclocomputador apresenta o problema que motivou este projeto: a indisponibilidade de inserção de um dado do usuário para a realização do cálculo de perda calórica. Portanto um teste comparativo não seria eficiente para verificar erros no protótipo.

O manual apresenta a seguinte tabela para exemplificação do consumo de calorias:

Tabela 4.6: Consumo Calórico no CATEYE VEL08

Km/h	10	20	30
Kcal por hora	67,3	244,5	641,6

No protótipo o consumo depende da massa, então para cada valor de massa colocado o resultado varia. Dentro dos parâmetros determinados para seu cálculo, a perda calórica funciona corretamente no protótipo.

Tabela 4.7: Consumo Calórico no Protótipo

Massa (Kg)	50	75	100
Kcal por hora	400	600	800

O valor MET utilizado é 8, que na tabela 2.1 da página 14 é considerado o valor geral para a atividade andar de bicicleta. Este valor também corresponde ao andar de bicicleta entre 19 e 22 km/h. Mesmo se fossem realizados testes nesta faixa de velocidade, o protótipo e o ciclocomputador de testes exibiriam valores diferentes, pois apenas no protótipo ocorre variação devido à massa do usuário.

5. CONSIDERAÇÕES FINAIS

Neste capítulo serão relatadas as dificuldades encontradas durante o desenvolvimento do projeto final (tanto da parte de *hardware* quanto de *software*), os resultados obtidos com o projeto, as propostas de melhorias que poderiam gerar novos projetos finais e por último a conclusão.

5.1. DIFICULDADES ENCONTRADAS

Foram encontradas várias dificuldades durante o desenvolvimento deste projeto, mais especificamente durante o desenvolvimento do protótipo.

Inicialmente o protótipo seria desenvolvido utilizando o *kit* do microcontrolador AT89S8252, mas durante os primeiros passos do desenvolvimento um problema foi identificado: como o desenvolvimento estava sendo em um *kit*, não haveria como trabalhar com cada pino separadamente utilizando os cabos de periféricos do conjunto, principalmente porque, como no exemplo do *display*, ele utiliza pinos tanto da *port* 1 como da *port* 3. Uma solução encontrada foi retirar o *chip* do microcontrolador e construir uma placa para o ciclocomputador, o que o tornaria mais compacto, uma vantagem para o protótipo. Mas outro problema veio logo em seguida: com o microcontrolador fora da placa do *kit*, não haveria como realizar a gravação. Gravadores para esse modelo de microcontrolador são caros, o que geraria um maior gasto com o projeto.

Outro ponto que levou à modificações na proposta original foi a dificuldade em trabalhar com a linguagem C. Também houve dificuldade durante a programação dos periféricos, que inicialmente seriam um *display LCD* e um teclado de 16 teclas. Especialmente a parte

referente ao teclado foi problemática, principalmente quando foi percebida a inutilidade de tantas teclas para o proposto no protótipo.

Neste ponto o protótipo teve de ser recomeçado praticamente do zero. Foi escolhido um novo microcontrolador, o AT89C2051. A sua escolha foi influenciada principalmente por ser um microcontrolador mais simples, com menos pinos, mais compacto e principalmente por ter se conseguido um gravador emprestado.

Além do microcontrolador, houve a definição dos componentes necessários para a placa. Foram utilizadas apenas duas teclas para inserção da massa, dispensando assim o teclado, e um sensor magnético, com atuação no *hardware* bem parecida com a dos botões. De acordo com as referências estudadas, fez-se necessário a utilização de resistores *de pull-ups* externos associados aos três, a fim de garantir o nível lógico dos pinos. Além desses, foram definidos a utilização de um circuito de oscilação, com a definição do cristal e capacitores, circuito de *reset*, o *display* e o controle de contraste, que até então não havia sido considerado, além do regulador de tensão de 5V, para garantir que não houvesse variação de tensão no protótipo

Foi escolhida a linguagem *Assembly* em substituição a C por ser uma linguagem mais utilizada para programação de microcontroladores, além de ser a linguagem usada quando a matéria de microcontroladores foi estudada no curso. Apesar de ser uma linguagem de baixo nível, trabalhar diretamente com espaços de memória tornou a compreensão do microcontrolador mais fácil e depois de passada a etapa de desenvolvimento da lógica e da programação, o algoritmo foi finalizado com sucesso.

Houve dificuldades durante o desenvolvimento da lógica e do algoritmo, pois era uma linguagem que o autor até então não tinha grande familiaridade. Durante a lógica, outro problema

do projeto foi revelado: as equações encontradas para se trabalhar já estavam definidas e eram corretas, mas não havia a possibilidade de implementá-las sem realizar adaptações matemáticas dentro da linguagem devido a duas restrições de hardware: o tamanho do espaço de memória, de 8 bits, e a falta de uma unidade de ponto flutuante.

Com registradores de 8 bits, cada um poderia armazenar no máximo um valor de 255, sendo que no transcorrer da utilização do microcontrolador suas saídas obteriam valores muito acima desses. Portanto uma das adaptações feitas foi o armazenamento em mais de um registrador, ora armazenando unidade, dezenas, centenas e milhares em registradores separados, ora utilizando dois registradores, com um armazenando a parte mais significativa e o outro a menos significativa; além de ter o cuidado de adaptar as fórmulas para que durante a sua execução não estourassem a capacidade dos registradores.

Outro ponto que levou às adaptações foi a falta da unidade de ponto flutuante no microcontrolador. Sem ela se tornou impossível realizar contas diretamente utilizando valores decimais. Portanto as equações não poderiam ser realizadas de modo convencional, foi necessário elaborar uma maneira para que elas gerassem tais valores apenas no último passo, fazendo com que o trabalho de armazenamento da parte decimal e da parte inteira, se tornasse mais simples e sem gerar grandes imprecisões de arredondamento.

Também vale ser mencionada a dificuldade no desenvolvimento das rotinas DIVISAO (DIVISAO_P_10 e DISISAO_P_100), as quais são trabalhadas principalmente a lógica binária, o que demandou um grande tempo até ser desenvolvida por completo.

5.2. RESULTADOS OBTIDOS

O protótipo do ciclocomputador é funcional, este é o maior resultado.

O *hardware* e o *software* funcionam como foram planejados, não havendo erros de funcionamento.

A distância é atualizada a cada ativação do sensor, possuindo erro de menos 2%, de acordo com os testes realizados.

A velocidade é atualizada a cada segundo, sendo exibida em múltiplos de 3,6, pois como a distância é obtida a cada metro percorrido e depois é transformada em km/h por esse fator de conversão, o resultado final fica atrelado a ele. Como ocorre a exibição apenas da parte inteira, o valor decimal é desprezado neste momento, apesar de ser contabilizado durante o cálculo.

A perda calórica é atualizada a cada 30 segundos de exercício ininterrupto, pois caso o usuário pare em algum momento durante o exercício, não faz sentido que a perda de calorias continue a ser contabilizada. O último valor permanece armazenado para então ser atualizado em 30 segundos depois do usuário voltar a pedalar.

5.3. SUGESTÃO DE TRABALHOS FUTUROS

Alguns projetos podem ser propostos para dar seqüência a este. Visando sempre a melhora da performance do ciclocomputador, pode-se propor as seguintes melhorias:

- Melhorar a obtenção da distância percorrida, reduzindo seu erro, além de permitir que o usuário possa realizar a escolha do aro da bicicleta, tornando assim o ciclocomputador utilizável em várias bicicletas.
- Buscar a exibição da velocidade a cada quilômetro por hora, permitindo que a sua atualização continue a cada segundo.
- Reduzir o tempo para a exibição da perda calórica.
- Realizar o cálculo da perda calórica utilizando o MET variando de acordo com a velocidade em cada instante, seguindo os valores para as velocidades mencionadas no Compêndio de Atividades Físicas, conseguindo assim uma perda calórica mais próxima ao real.
- Acrescentar novas informações a serem exibidas ao usuário, como por exemplo, o tempo de exercício, opção de cronometragem e velocidade máxima.

5.4. CONCLUSÃO

O objetivo deste projeto final, que foi o desenvolvimento de um protótipo de ciclocomputador que, através da inserção da massa do usuário, retornasse a ele seu gasto energético, além de exibir a distância percorrida e a velocidade instantânea, foi alcançado, e com isso, a proposta de projeto final apresentada foi cumprida. A precisão atual é considerada satisfatória para um protótipo acadêmico e é algo que pode ser melhorada em projetos futuros que sigam este inicial.

Durante o desenvolvimento deste trabalho foi possível aprofundar os conhecimentos estudados durante todo o curso, principalmente os fundamentos de microcontroladores, lógica e programação, além de adquirir novos conhecimentos, relativos à fisiologia. E com isso se pode concluir que o agregamento de diversos conhecimentos para um engenheiro é essencial para sua educação, pois não será sempre que ele terá de resolver problemas apenas dentro da sua área de conhecimento, quiçá sendo necessário buscar conhecimentos em outras áreas para a solução de problemas.

6. REFERÊNCIAS BIBLIOGRÁFICAS

AMORIM, Paulo R.; GOMES, Thales N. P. *Gasto Energético na Atividade Física*. Rio de Janeiro: Shape, 2003. Cap.5.

FOSS, Merle L.; KETEYIAN, Steven J. Bases *Fisiológicas do Exercício e do Esporte*. 6.ed. Rio de Janeiro: Guanabara Koogan, 2000. Cap.4. Ap. D, Ap. G.

JUNIOR, Vidal P. S. *Práticas do Microcontrolador 8051*. 7. Ed. São Paulo: Érica, 1998.

MCARDLE, William D. ET AL. *Fundamentos de Fisiologia do Exercício*. 2.ed. Rio de Janeiro: Guanabara Koogan, [2002]. Cap.6, Cap.7.

NICOLOSI, Denys E. C. *Microcontrolador 8051 Detalhado*. 5.ed. São Paulo: Érica, 2004.

NUSSENZVEIG, H. Moisés. *Curso de Física Básica Volume 1*, 2.ed. São Paulo: Edgard Blücher. Cap.2, Cap.6, Cap.7

NUSSENZVEIG, H. Moisés. *Curso de Física Básica Volume 2*, 3.ed. São Paulo: Edgard Blücher. Cap.7, Cap.8.

PINI, Mário C. *Fisiologia Esportiva*. Rio de Janeiro: Guanabara Koogan, 1978. Cap.8.

ROBERGS, Robert A.; ROBERTS, Scott O. *Princípios Fundamentais de Fisiologia do Exercício...* 1.ed. São Paulo: Phorte, 2002. Cap.4, Ap. D, Ap. E.

TIPLER, Paul A. *Física Volume 1*. 4.ed. Rio de Janeiro: Livros Técnicos e Científicos, 2000. Cap.2, Cap.6

57ª ASSEMBLÉIA MUNDIAL DE SAÚDE. *Estratégia Global em Alimentação Saudável*, Atividade Física e Saúde. 2004. Disponível em: <http://dtr2004.saude.gov.br/nutricao/documentos/eb_portugues.pdf>. Acesso em: 20 ago. 2007.

ABRACICLO, Associação Brasileira dos Fabricantes de Motocicletas, Ciclomotores, Motonetas, Bicicletas e Similares. Bicicletas. São Paulo. Disponível em: <http://www.abraciclo.com.br/> Acesso em 15 set. 2007.

AINSWORTH BE. *The Compendium of Physical Activities Tracking Guide*. Prevention Research Center, Norman J. Arnold School of Public Health, University of South Carolina, 2002. Disponível em: <http://prevention.sph.sc.edu/tools/docs/documents_compendium.pdf> Acesso em 09 ago. 2007.

ALMEIDA, et al. *Gasto Calórico nas Atividades de Trabalho e Cotidianas, dos Carteiros que Utilizam Bicicleta*. Revista Brasileira de Cineantropometria e Desempenho Humano, Volume 6 Número 2. 2004. p.53-61. Disponível em: <http://www.rbcdh.ufsc.br/DownloadArtigo.do;jsessionid=625126356051A973F05CB97349C3155D?artigo=201> Acesso em 06 ago. 2007.

ATMEL Corporation. *AT89C2051*. 2005. Disponível em: <http://www.atmel.com/atmel/acrobat/doc0368.pdf> Acesso em: 05 abr. 2008.

BARBACENA, Ilton L.; FLEURY, Cláudio A. *Display LCD*. 1996. Disponível em: <http://www2.eletronica.org/apostilas-e-ebooks/componentes/LCD_30324b.pdf> Acesso em: 10 mai. 2008

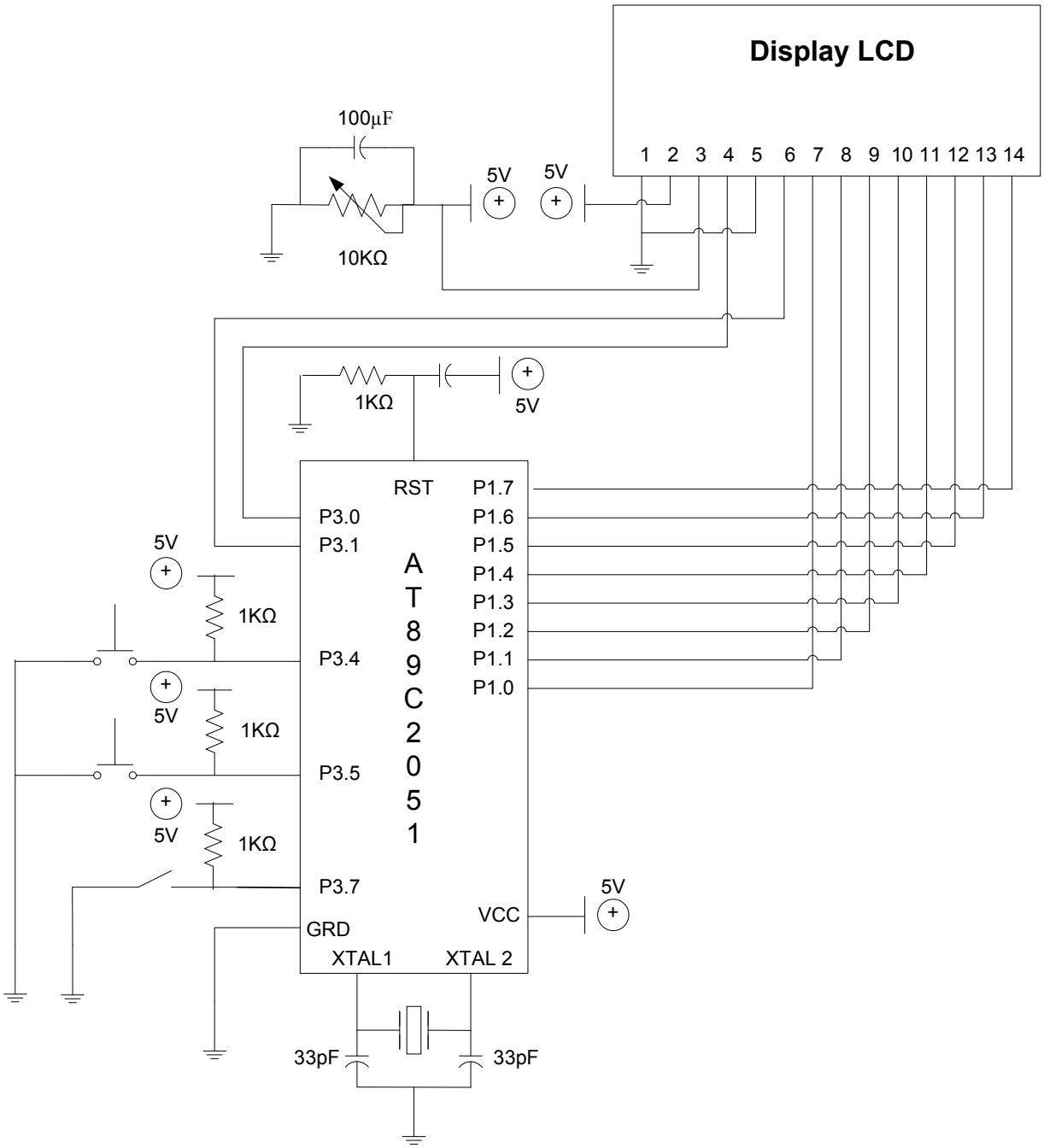
CALOI. *História das bicicletas*. Disponível em: <http://www.caloi.com/> Acesso em: 15 set. 2007.

CDOF, Cooperativa do Fitness. Ciclismo. Belo Horizonte. Disponível em: <http://www.cdof.com.br/ciclismo.htm> Acesso em 21 ago. 2007.

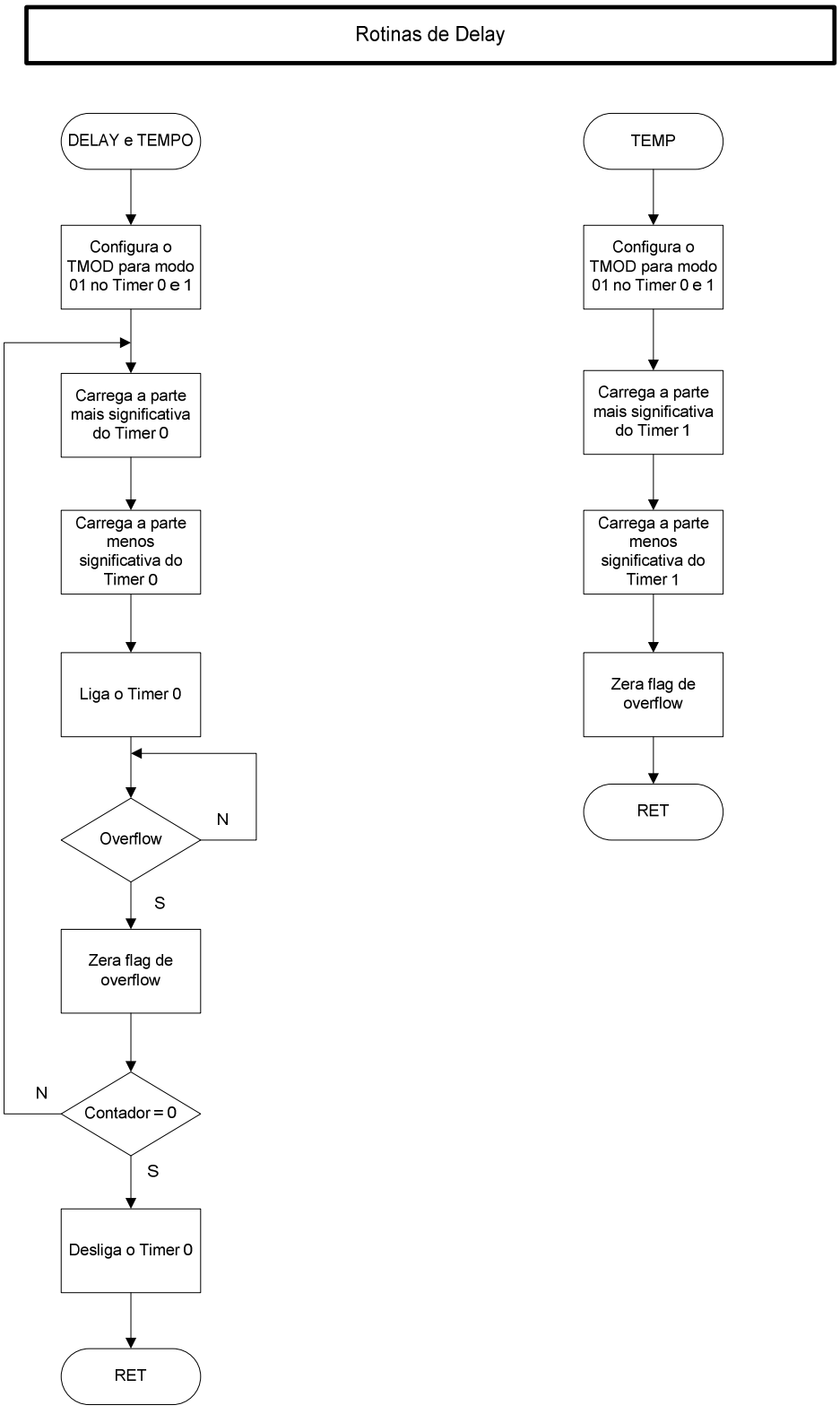
RICARTE, Ivan L. M. *Programação de Sistemas*: uma introdução. 2000. Disponível em: <http://www.dca.fee.unicamp.br/cursos/EA876/apostila/> Acesso em: 15 mar. 2008.

RUNNING BR, *Treinamento na Esteira*. 2006. Disponível em: <http://www.pntreinamento.com.br/site/pn_noticias/index.jsp?cod=172>. Acesso em: 23 set. 2007.

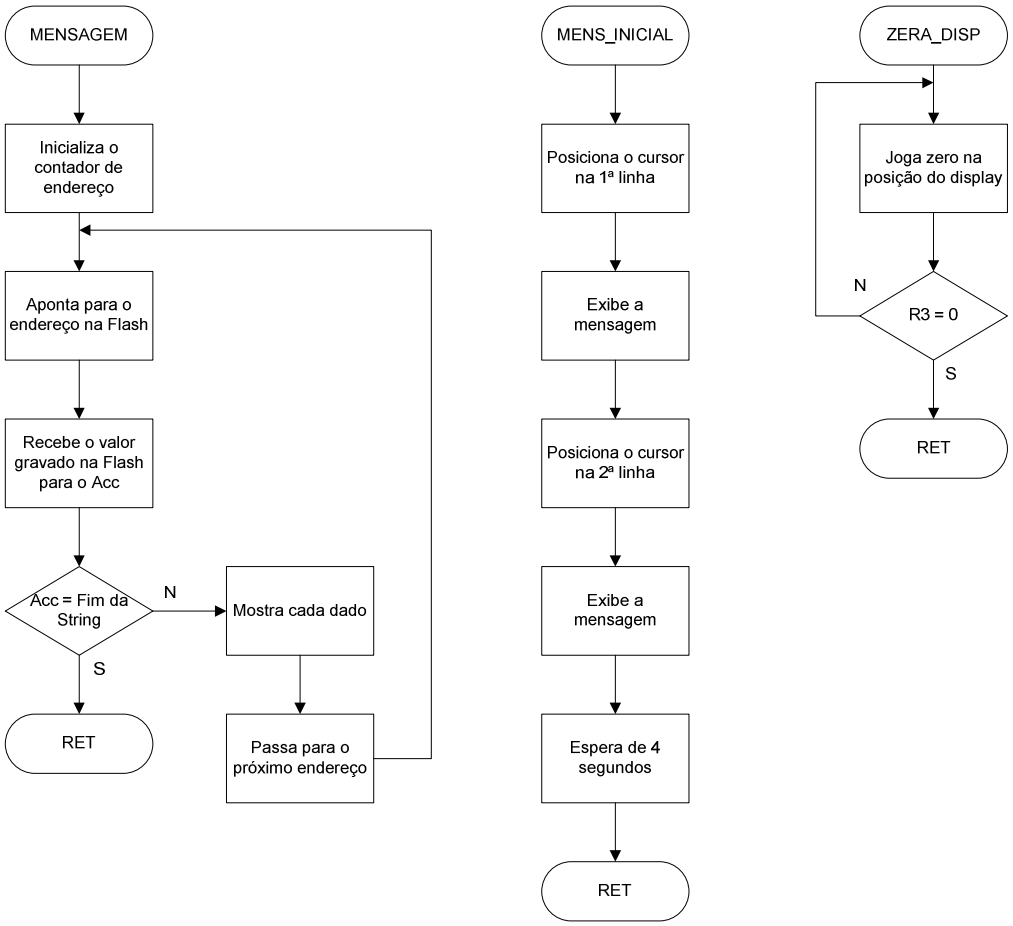
APÊNDICE A – *DATASHEET* DO CICLOCOMPUTADOR

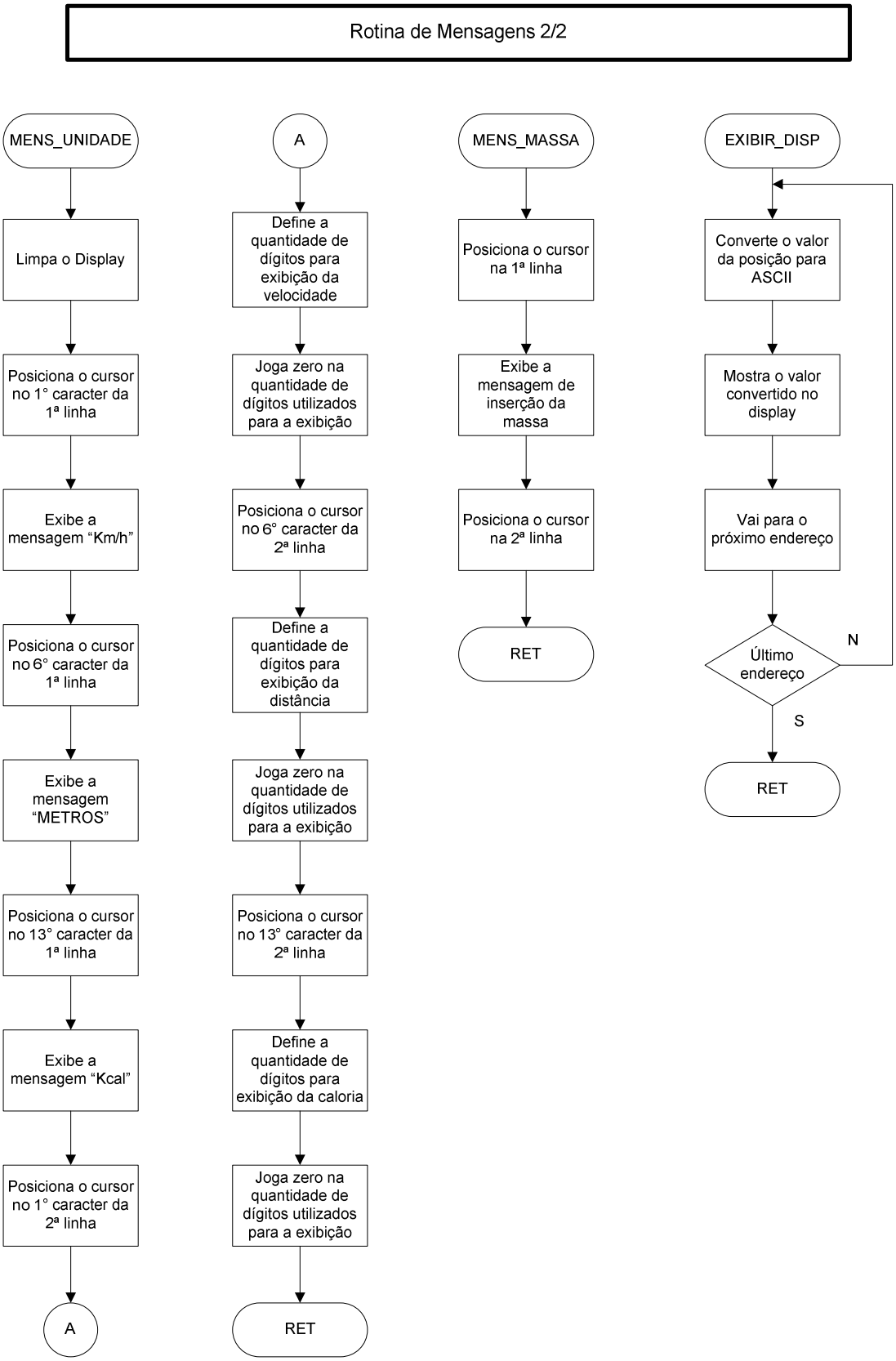


APÊNDICE B – FLUXOGRAMAS

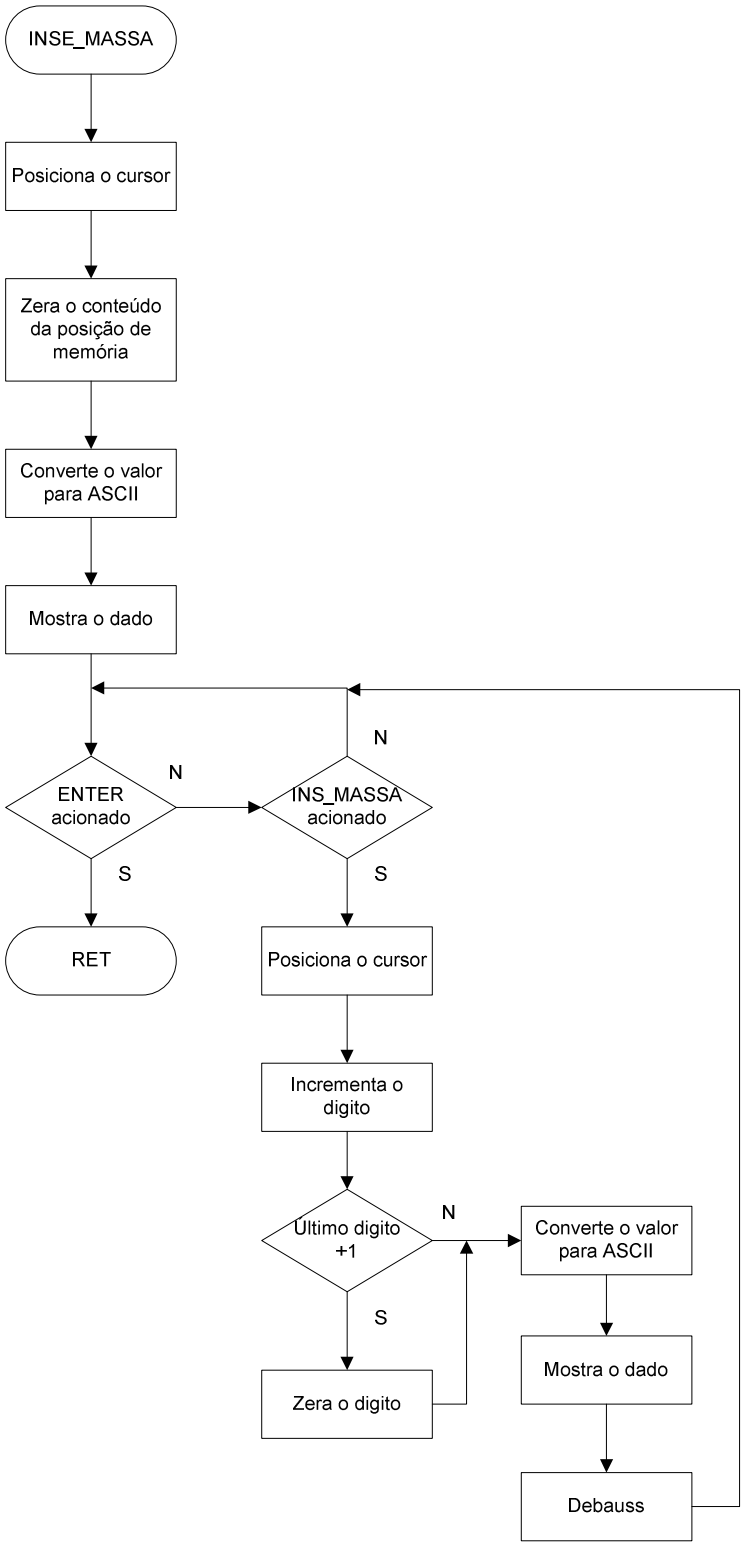


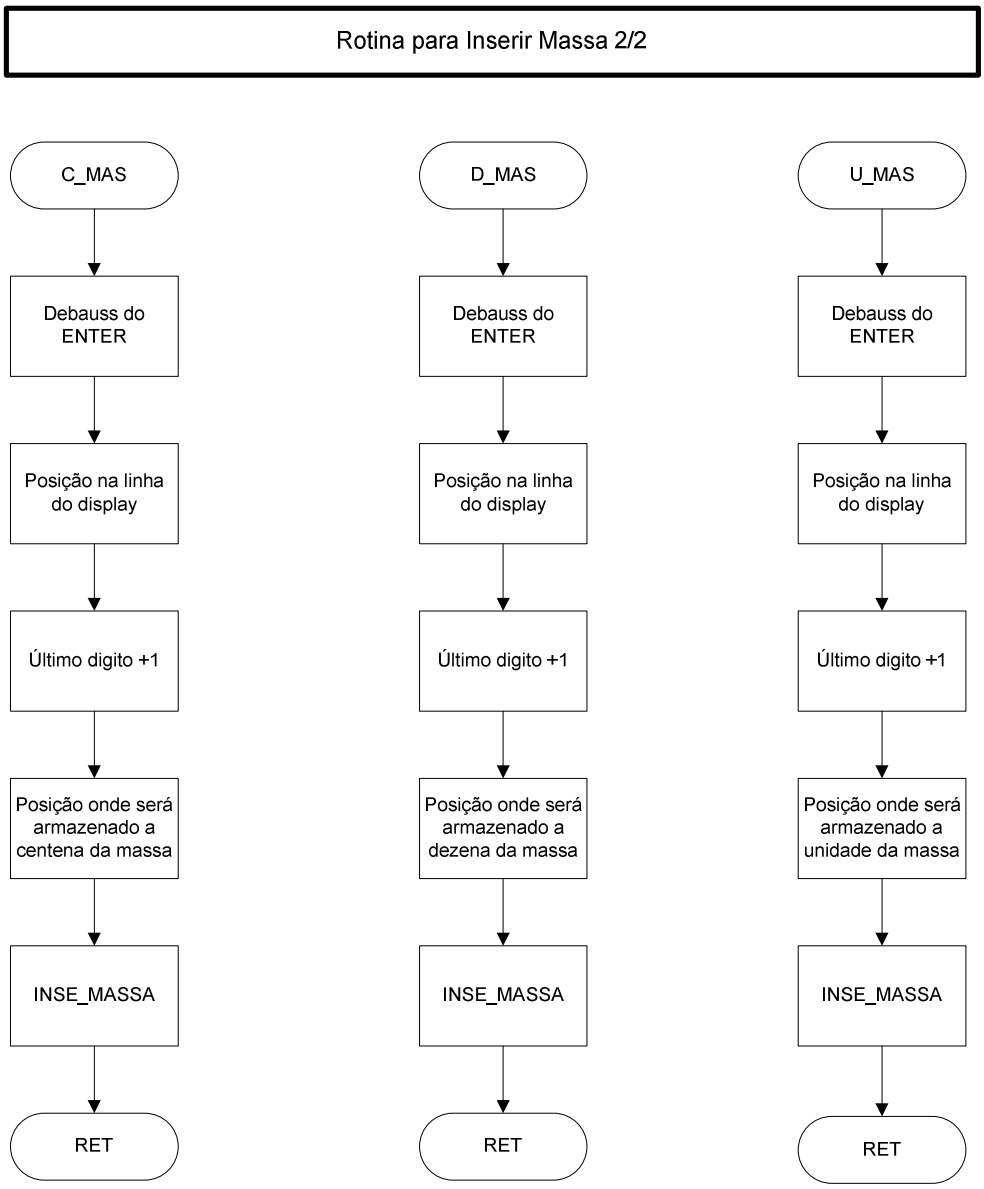
Rotina de Mensagens 1/2



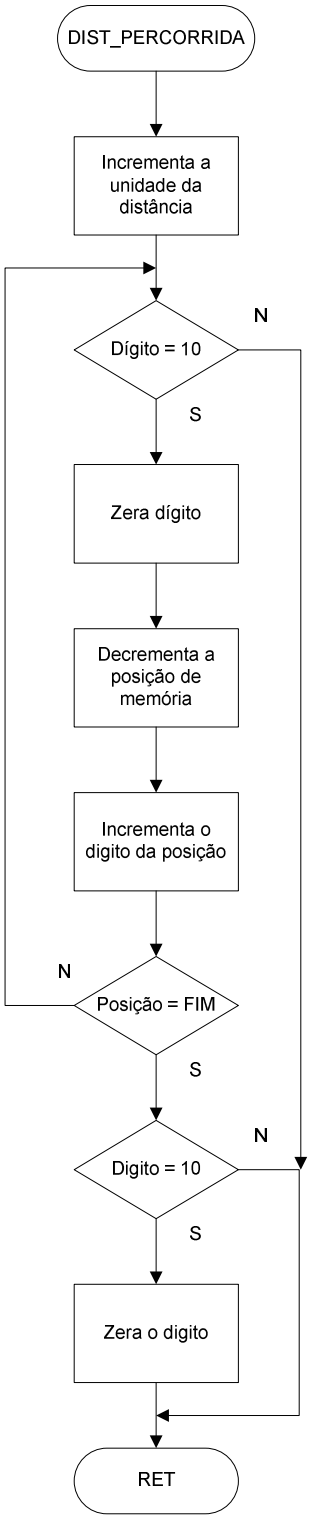

Rotina de Mensagens 2/2
MENS_UNIDADE
Limpa o Display
Posiciona o cursor no 1° caracter da 1ª linha
Exibe a mensagem “Km/h”
Posiciona o cursor no 6° caracter da 1ª linha
Exibe a mensagem “METROS”
Posiciona o cursor no 13° caracter da 1ª linha
Exibe a mensagem “Kcal”
Posiciona o cursor no 1° caracter da 2ª linha
A
A
Define a quantidade de dígitos para exibição da velocidade
Joga zero na quantidade de dígitos utilizados para a exibição
Posiciona o cursor no 6° caracter da 2ª linha
Define a quantidade de dígitos para exibição da distância
Joga zero na quantidade de dígitos utilizados para a exibição
Posiciona o cursor no 13° caracter da 2ª linha
Define a quantidade de dígitos para exibição da caloria
Joga zero na quantidade de dígitos utilizados para a exibição
RET
MENS_MASSA
Posiciona o cursor na 1ª linha
Exibe a mensagem de inserção da massa
Posiciona o cursor na 2ª linha
RET
EXIBIR_DISP
Converte o valor da posição para ASCII
Mostra o valor convertido no display
Vai para o próximo endereço
Último endereço
N
S
RET

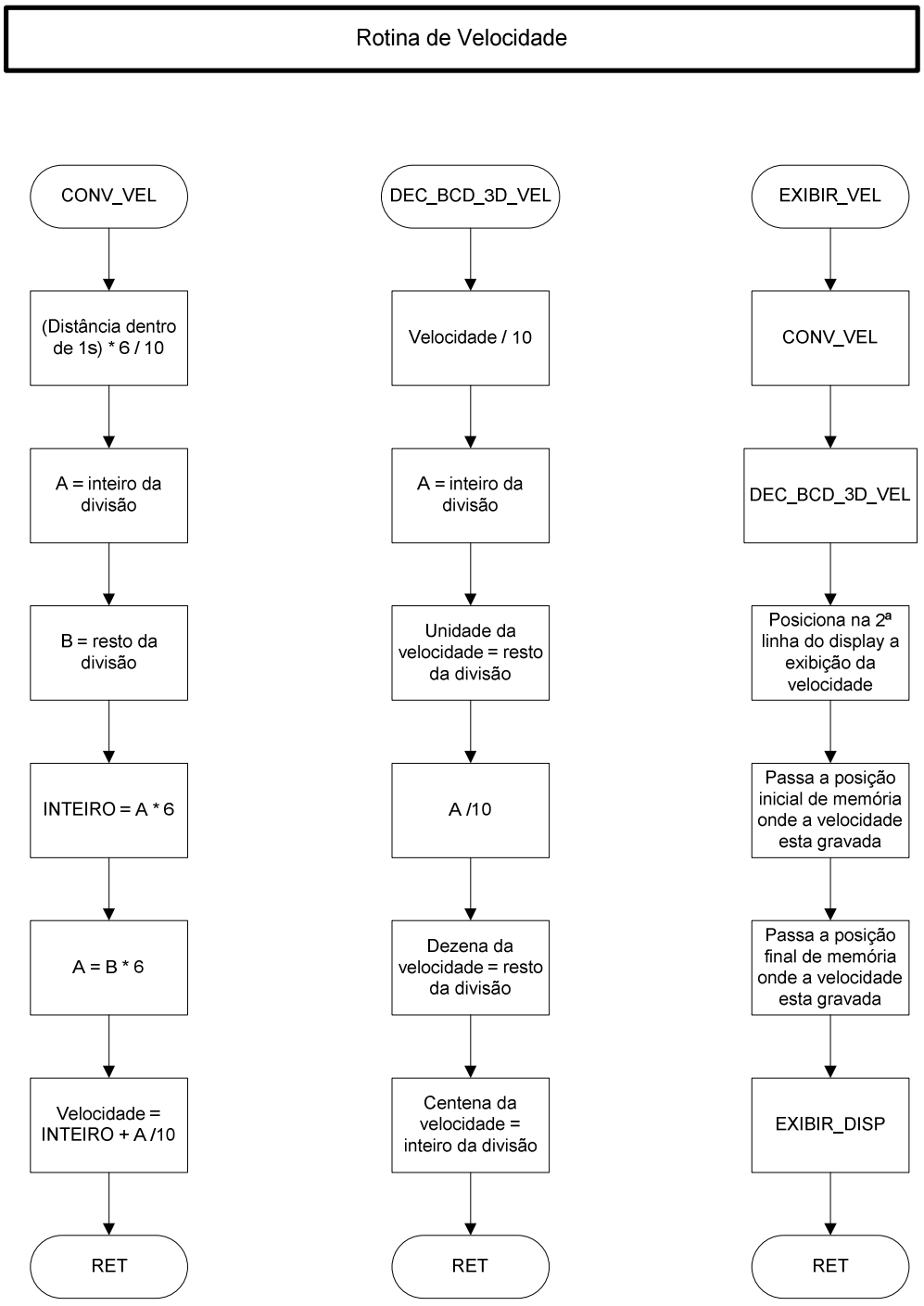
Rotina para Inserir Massa 1/2



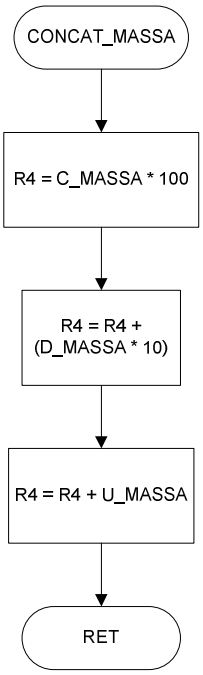

Rotina para Inserir Massa 2/2
C_MAS
Debauss do ENTER
Posição na linha do display
Último digito +1
Posição onde será armazenado a centena da massa
INSE_MASSA
RET
D_MAS
Debauss do ENTER
Posição na linha do display
Último digito +1
Posição onde será armazenado a dezena da massa
INSE_MASSA
RET
U_MAS
Debauss do ENTER
Posição na linha do display
Último digito +1
Posição onde será armazenado a unidade da massa
INSE_MASSA
RET

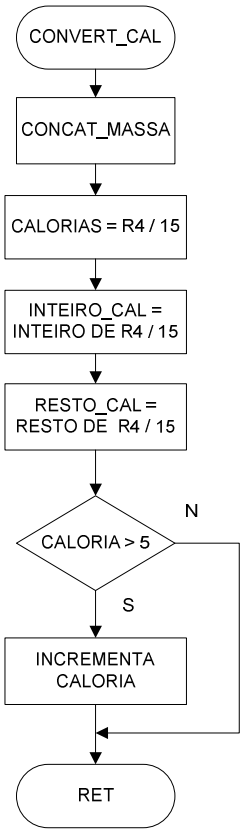
Rotina de Distância

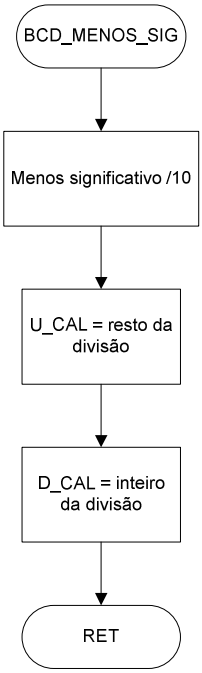


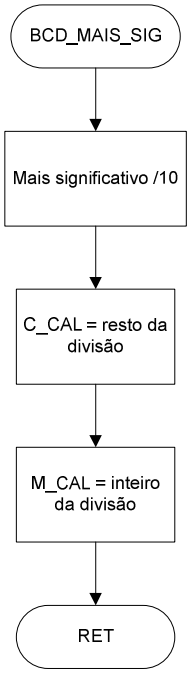

Rotina de Velocidade
CONV_VEL
(Distância dentro de 1s) * 6 / 10
A = inteiro da divisão
B = resto da divisão
INTEIRO = A * 6
A = B * 6
Velocidade = INTEIRO + A /10
RET
DEC_BCD_3D_VEL
Velocidade / 10
A = inteiro da divisão
Unidade da velocidade = resto da divisão
A /10
Dezena da velocidade = resto da divisão
Centena da velocidade = inteiro da divisão
RET
EXIBIR_VEL
CONV_VEL
DEC_BCD_3D_VEL
Posiciona na 2ª linha do display a exibição da velocidade
Passa a posição inicial de memória onde a velocidade esta gravada
Passa a posição final de memória onde a velocidade esta gravada
EXIBIR_DISP
RET

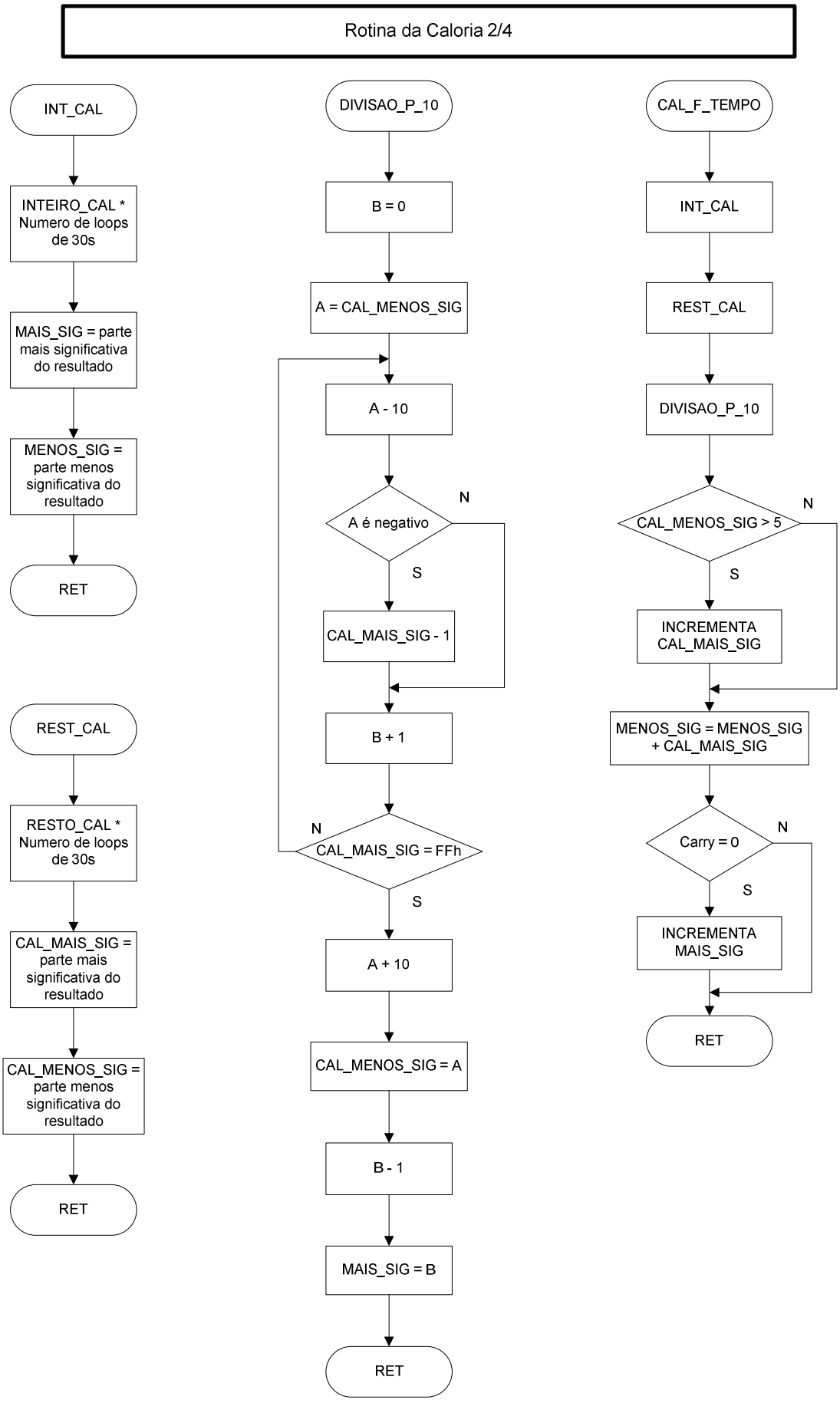
Rotina da Caloria 1/4



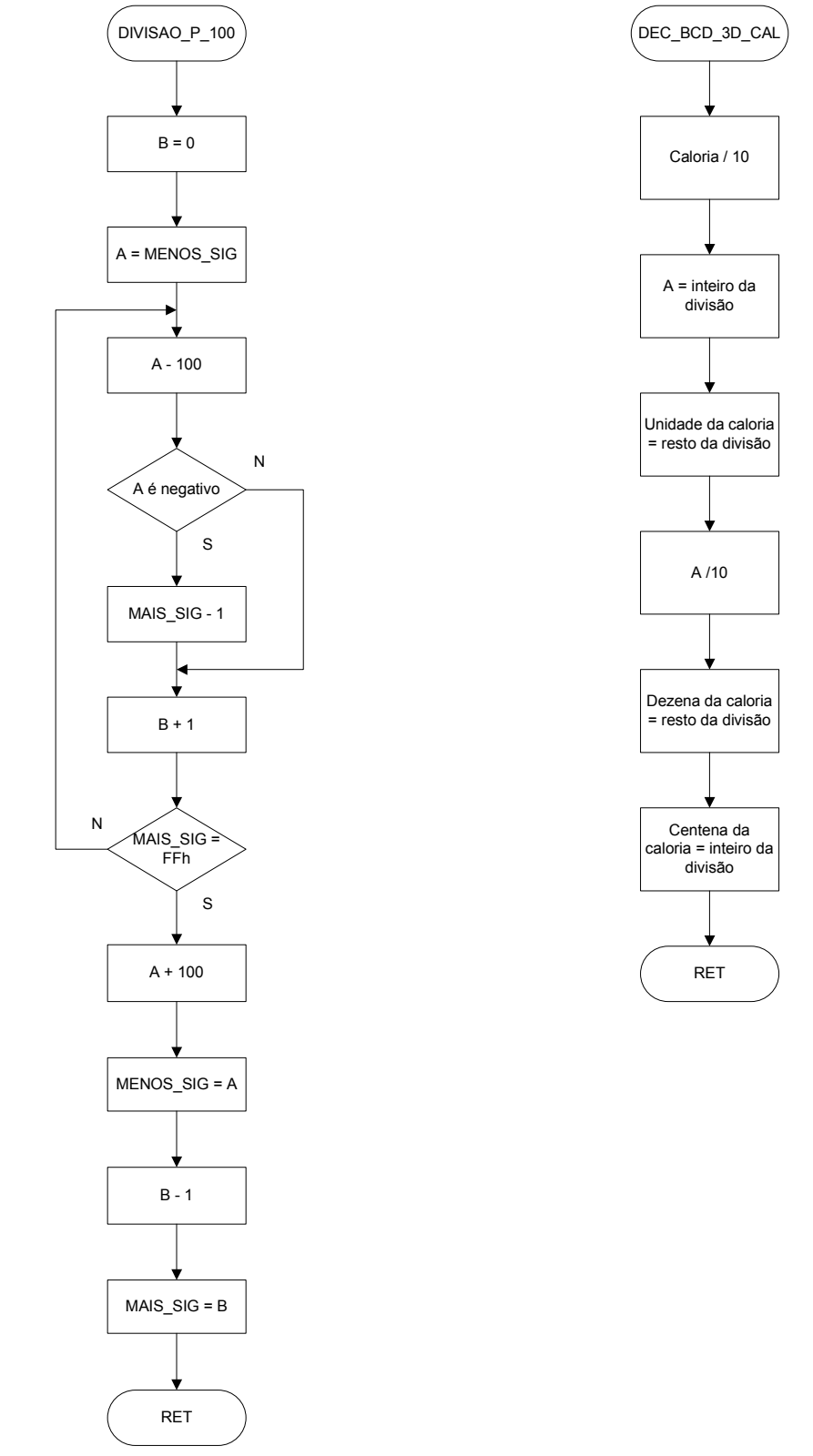




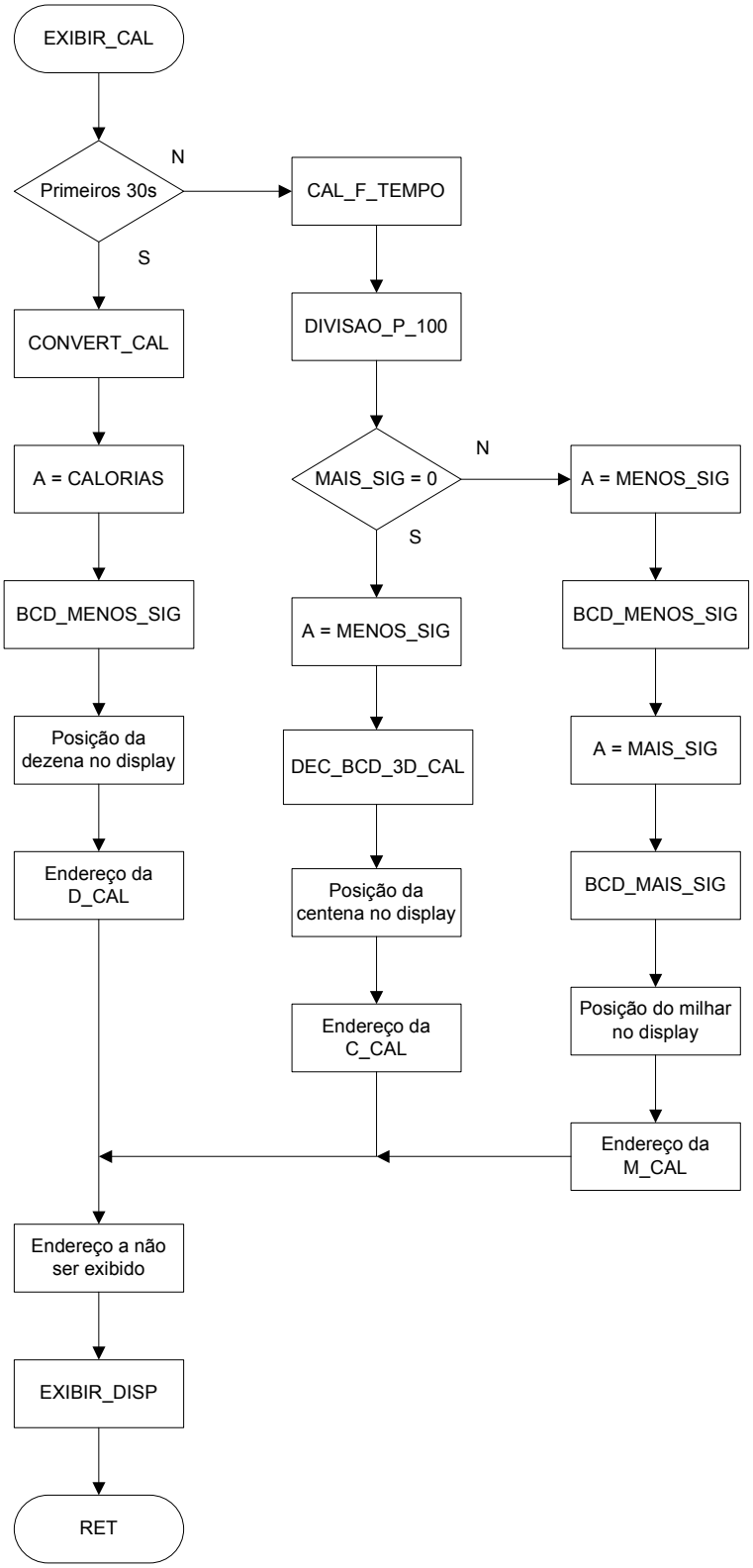



Rotina da Caloria 2/4
INT_CAL
INTEIRO_CAL * Numero de loops de 30s
MAIS_SIG = parte mais significativa do resultado
MENOS_SIG = parte menos significativa do resultado
RET
REST_CAL
RESTO_CAL * Numero de loops de 30s
CAL_MAIS_SIG = parte mais significativa do resultado
CAL_MENOS_SIG = parte menos significativa do resultado
RET
DIVISAO_P_10
B = 0
A = CAL_MENOS_SIG
A - 10
A é negativo
N
S
CAL_MAIS_SIG - 1
B + 1
CAL_MAIS_SIG = FFh
N
S
A + 10
CAL_MENOS_SIG = A
B - 1
MAIS_SIG = B
RET
CAL_F_TEMPO
INT_CAL
REST_CAL
DIVISAO_P_10
CAL_MENOS_SIG > 5
N
S
INCREMENTA CAL_MAIS_SIG
MENOS_SIG = MENOS_SIG + CAL_MAIS_SIG
Carry = 0
N
S
INCREMENTA MAIS_SIG
RET

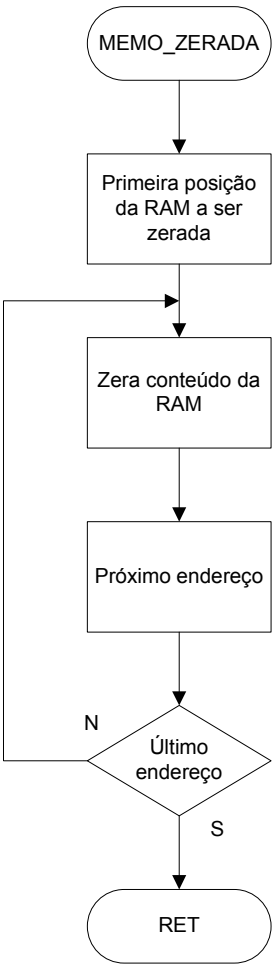
Rotina da Caloria 3/4

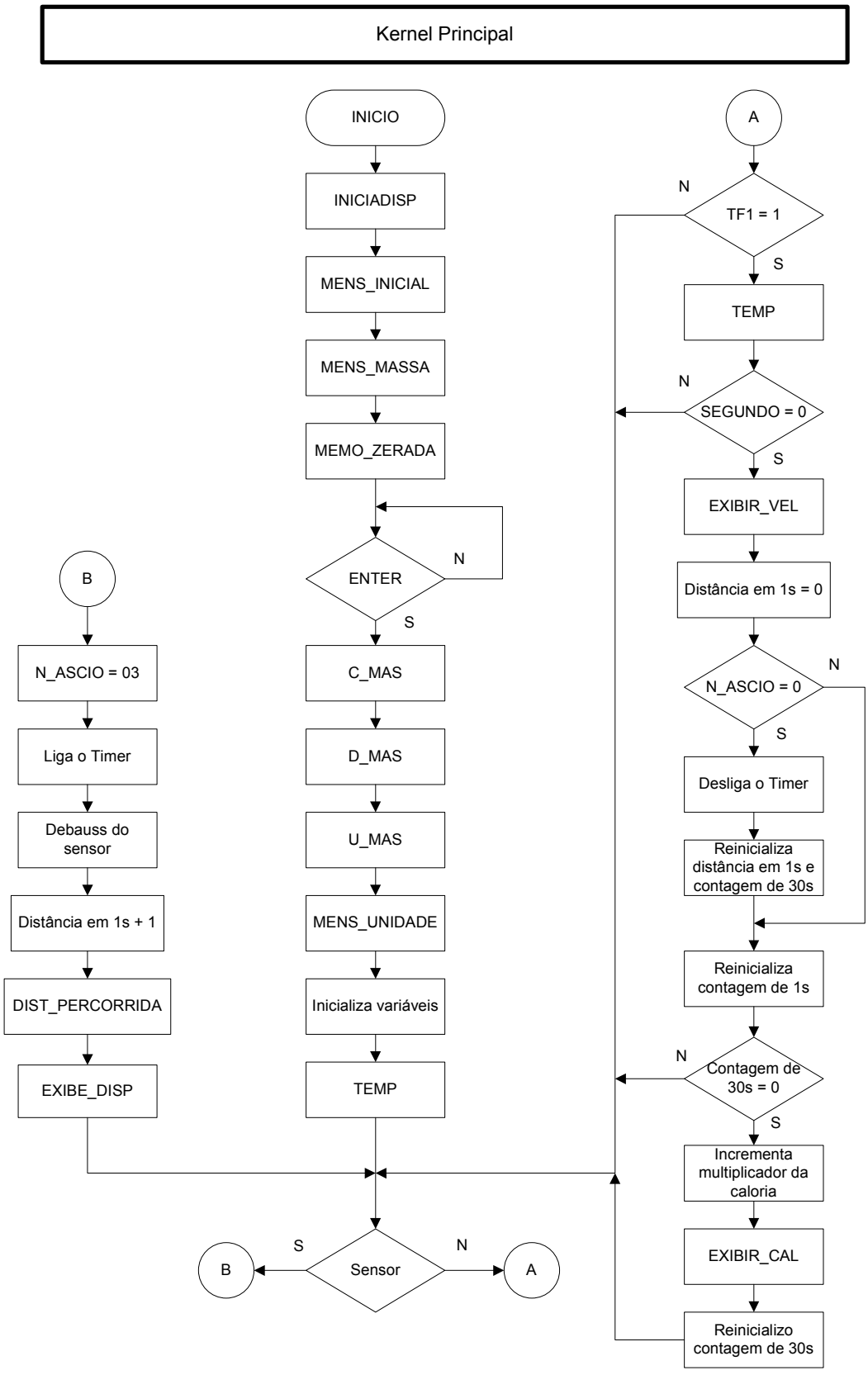


Rotina da Caloria 4/4



Rotina para Limpar Memória




Kernel Principal
INICIO
INICIADISP
MENS_INICIAL
MENS_MASSA
MEMO_ZERADA
ENTER
N
S
C_MAS
D_MAS
U_MAS
MENS_UNIDADE
Inicializa variáveis
TEMP
B
N_ASCIO = 03
Liga o Timer
Debauss do sensor
Distância em 1s + 1
DIST_PERCORRIDA
EXIBE_DISP
Sensor
S
N
B
A
A
TF1 = 1
N
S
TEMP
SEGUNDO = 0
N
S
EXIBIR_VEL
Distância em 1s = 0
N_ASCIO = 0
N
S
Desliga o Timer
Reinicializa distância em 1s e contagem de 30s
Reinicializa contagem de 1s
Contagem de 30s = 0
N
S
Incrementa multiplicador da caloria
EXIBIR_CAL
Reinicializo contagem de 30s

APÊNDICE C – ALGORÍTIMO

```
$SUBTITLE(CICLO COMPUTADOR.ASM)
$PG PL=60
$PG PAGEWIDTH=120
$ALLPUBLIC
$PAGINATE


      DEFSEG  ABSSEGNAME, ABSOLUTE
         SEG     ABSSEGNAME

HABDADO          EQU     P3.0
HAB              EQU     P3.1
ENTER            EQU     P3.4
INS_MASSA        EQU     P3.5
SENSOR           EQU     P3.7
F_STR            EQU     '$'

C_VEL            EQU     27H
D_VEL            EQU     28H
U_VEL            EQU     29H
C_MASSA          EQU     2AH
D_MASSA          EQU     2BH
U_MASSA          EQU     2CH
M_CAL            EQU     2DH
C_CAL            EQU     2EH
D_CAL            EQU     2FH
U_CAL            EQU     30H
IN_MEMO          EQU     31H
N_ASCIO          EQU     32H
RESTO            EQU     33H
INTEIRO          EQU     34H
INTEIRO_CAL      EQU     35H
MENOS_SIG        EQU     38H
MAIS_SIG         EQU     39H
DIG_MAX          EQU     3AH
POS_DISP         EQU     3BH
SEGUNDO          EQU     3CH
FIM_EXIB         EQU     3DH
RESTO_CAL        EQU     40H
CALORIAS         EQU     41H
CAL_MAIS_SIG     EQU     42H
CAL_MENOS_SIG    EQU     43H


                         ORG 0000H
                         LJMP INICIO             ;SALTA PARA A ROTINA
INICIO


MENS                     DB       'CICLO COMPUTADOR$'
VER                      DB       'VERSAO 1.0 $'
MASSA                    DB       'INSIRA A MASSA:$'
VELOCIDADE               DB       'Km/h$'
```

```
DISTANCIA               DB      'METROS$'
CALORIA                 DB      'KCAL$'

;######################ROTINAS DE DELAY#########################################

DELAY:
            MOV   TMOD,#$11         ;PROGRAMACAO DO MODO 01 NO TIMER 0
LOOP:
            MOV   TH0,#$FC          ;NESSE DOIS REGISTRADORES SAO CARR;
            MOV   TL0,#$17          ;O VAL INI DE 64535 PARA CONTAR 1000us
            SETB  TR0               ;LIGA O TIMER
            JNB   TF0,$             ;VERIFICA O FIM DA CONTAGEM
            CLR   TF0               ;ZERA A FLAG QUE INDICA ESTOURO
            DJNZ  R2,LOOP           ;DIFINE O NUMERO DE TEMPO
            CLR   TR0               ;DESLIGA O TIMER
            RET

TEMPO:
            MOV   TMOD,#$11         ;PROGRAMACAO DO MODO 01 NO TIMER 0
LOOP_1:
            MOV   TH0,#$0B          ;NESSE DOIS REGISTRADORES SAO CARREGADOS
            MOV   TL0,#$DB          ;O VALOR INICIAL 3035 PARA CONTAR 62500us
            SETB  TR0               ;LIGA O TIMER
            JNB   TF0,$             ;VERIFICA O FIM DA CONTAGEM
            CLR   TF0               ;ZERA A FLAG QUE INDICA ESTOURO
            DJNZ  R2,LOOP_1         ;DIFINE O NUMERO DE TEMPO
            CLR   TR0               ;DESLIGA O TIMER
            RET

TEMP:
            MOV   TMOD,#$11         ;PROGRAMACAO DO MODO 01 NO TIMER 1
            MOV   TH1,#$0B          ;NESSE DOIS REGISTRADORES SAO CARREGADOS;
            MOV   TL1,#$DB          ;O VALOR INICIAL 3035 PARA CONTAR 62500us
            CLR   TF1               ;ZERA O OVERFLOW
            RET


;##############################ROTINAS DE DISPLAY################################

DADO:
            SETB   HABDISP        ;HABILITA DISPLAY
            SETB   HABDADO        ;PREPARA O DISPLAY PARA RECEBER DADOS
            MOV    R2,#$01        ;DEFINE A QUANTIDADE DE TEMPO
            LCALL  DELAY          ;RETARDO PARA CONFIGURACAO DO DADO
            CLR    HABDISP        ;DESABILITA DISPLAY
            RET                   ;FIM DE ROTINA

INSTRUCAO:
            SETB   HABDISP        ;HABILITA DISPLAY
            CLR    HABDADO        ;PREPARA DISPLAY PARA RECEBER INSTRUCAO
            MOV    R2,#$03        ;DEFINE A QUANTIDADE DE TEMPO
            LCALL  DELAY          ;RETARDO PARA CONFIGURACAO DE INSTRUCAO
            CLR    HABDISP        ;DESABILITA DISPLAY
            RET                   ;FIM DE ROTINA

INICIADISP:
            MOV    P1,#$38        ;CONFIGURA DISPLAY EM MODO 8 BITS
```

```
            LCALL  INSTRUCAO      ;ROTINA DE HABILITACAO DE INSTRUCAO
            MOV    P1,#$0C        ;CONFIGURA DISPLAY ACESO COM CURSOR APAGADO
            LCALL  INSTRUCAO      ;ROTINA DE HABILITACAO DE INSTRUCAO
            MOV    P1,#$06        ;CONFIG DISP P ESCREVER C/ CURSOR P DIREITA
            LCALL  INSTRUCAO      ;ROTINA DE HABILITACAO DE INSTRUCAO
            MOV    P1,#$01        ;CONFIGURA DISPLAY PARA LIMPAR
            LCALL  INSTRUCAO      ;ROTINA DE HABILITACAO DE INSTUCAO
            RET                   ;FIM DE ROTINA

;######################ROTINAS DE MENSAGENS ###################################

MENSAGEM:
            MOV    B,#$00          ;END JUNTO AO DPTR E RELACIONA MEMO DE
PROG
VOLTA:
            MOV    A,B             ;APONTA A POSICAO NA MEMO DE PROG
            MOVC   A,@A+DPTR       ;PEGA O VALOR DA POSICAO APONTADA
            CJNE   A,#F_STR,RECMEN ;VERIFICA SE E O FIM DA MENSAGEM
            RET
RECMEN:
            MOV    P1,A            ;MOSTRA CADA LETRA
            LCALL  DADO            ;IMPRIME OS DADOS
            INC    B               ;VAI PARA O PROXIMO ENDERECO
            LJMP   VOLTA           ;BUSCA A PROXIMA LETRA


MENS_INICIAL:
            MOV     P1,#$80       ;POSICAO DA MENSAGEM
            LCALL   INSTRUCAO     ;HABILITA A INSTRUCAO
            MOV     DPTR,#MENS    ;MENSAGEM CICLO COMPUTADOR
            LCALL   MENSAGEM      ;MOSTRA A MENSAGEM
            MOV     P1,#$C2       ;POSICAO DA MENSAGEM
            LCALL   INSTRUCAO     ;HABILITA A INSTRUCAO
            MOV     DPTR,#VER     ;MENSAGEM VERSAO 01
            LCALL   MENSAGEM      ;MOSTRA A MENSAGEM
            MOV     R2,#$40       ;DEFINE O TEMPO EM 4 SEG
            LCALL   TEMPO         ;TEMPORIZADOR
            RET

ZERA_DISP:
            MOV   P1,#$30         ;CODIGO ASCII DO 0
            LCALL DADO            ;ROTINA QUE MOSTRA DADO NO DISPLAY
            DJNZ  R3,ZERA_DISP    ;QUANTIDADE DE DIGITOS PARA SER ZERADO
            RET

MENS_UNIDADE:
            MOV   P1,#$01
            LCALL INSTRUCAO

            MOV   P1,#$80         ;POSICAO NA LINHA DO DISPLAY
            LCALL INSTRUCAO       ;ROTINA QUE PASSA INSTRUCAO PARA O DISPLAY
            MOV   DPTR,#VELOCIDADE;MENSAGE KM/H
            LCALL MENSAGEM        ;ROTINA QUE PASSA MENSAGEM PARA O DISPLAY

            MOV   P1,#$85         ;POSICAO NA LINHA DO DISPLAY
            LCALL INSTRUCAO       ;ROTINA QUE PASSA INSTRUCAO PARA O DISPLAY
            MOV   DPTR,#DISTANCIA ;MENSAGE METROS
```

```
                LCALL MENSAGEM          ;ROTINA QUE PASSA MENSAGEM PARA O DISPLAY

                MOV   P1,#$8C           ;POSICAO NA LINHA DO DISPLAY
                LCALL INSTRUCAO         ;ROTINA QUE PASSA INSTRUCAO PARA O DISPLAY
                MOV   DPTR,#CALORIA     ;MENSAGEM KCAL
                LCALL MENSAGEM          ;ROTINA QUE PASSA MENSAGEM PARA O DISPLAY

                MOV   P1,#$C0           ;POSICAO NO DISPLAY INICIAL DA VELOCIDADE
                LCALL INSTRUCAO         ;ROTINA DE INSTRUCAO NO DISPLAY
                MOV   R3,#$03           ;QUANT DE DIGITOS P MOSTRAR ZERO NO DISPLAY
                CALL  ZERA_DISP         ;ROTINA PARA JOGAR ZERO NA POSICAO INDICADA

                MOV   P1,#$C5           ;POSICAO NO DISPLAY INICIAL DA DISTANCIA
                LCALL INSTRUCAO         ;ROTINA DE INSTRUCAO NO DISPLAY
                MOV   R3,#$06           ;QUANT DE DIGITOS P MOSTRAR ZERO NO DISPLAY
                CALL  ZERA_DISP         ;ROTINA PARA JOGAR ZERO NA POSICAO INDICADA

                MOV   P1,#$CC           ;POSICAO NO DISPLAY INICIAL DA KCAL
                LCALL INSTRUCAO         ;ROTINA DE INSTRUCAO NO DISPLAY
                MOV   R3,#$04           ;QUANT DE DIGITOS P MOSTRAR ZERO NO DISPLAY
                CALL  ZERA_DISP         ;ROTINA PARA JOGAR ZERO NA POSICAO INDICADA
                RET

MENS_MASSA:
                MOV   P1,#$81           ;POSICAO NA LINHA DO DISPLAY
                LCALL INSTRUCAO         ;ROTINA QUE PASSA INSTRUCAO PARA O DISPLAY
                MOV   DPTR,#MASSA       ;MENSAGEM INSIRA A MASSA
                LCALL MENSAGEM          ;ROTINA QUE PASSA MENSAGEM PARA O DISPLAY
                MOV   P1,#$C1           ;POSICAO NA LINHA DO DISPLAY
                LCALL INSTRUCAO         ;ROTINA QUE PASSA INSTRUCAO PARA O DISPLAY
                RET

EXIBIR_DISP:
                MOV   A,@R0             ;VALOR PARA CONVERTER PARA O DISPLAY
                ADD   A,#$30            ;CONVERTE EM ASCII PARA JOGAR NO DISPLAY
                MOV   P1,A              ;VALOR PARA APRESENTAR NO DISPLAY
                LCALL DADO              ;PASSA VALORES PARA O DISPLAY
                INC   R0                ;VAI PARA O PROXIMO ENDERECO
                MOV   A,R0              ;BUSCAR O ULTIMO ENDERECO
                CJNE  A,FIM_EXIB,EXIBIR_DISP ;TESTA SE E O ULTIMO DISPLAY
                RET

;#####################ROTINAS DE INSERIR MASSA#################################

INSE_MASSA:
                MOV   P1,POS_DISP       ;POSICAO NO DISPLAY
                LCALL INSTRUCAO         ;PASSA INTRUCAO PARA O DISPLAY
                MOV   A,@R0
                ADD   A,#$30
                MOV   P1,A              ;VALOR EM ASCII PARA O DISPLAY
                LCALL DADO              ;MOSTRA VALOR NO DISPLAY
INC_PR:
                JNB   ENTER,PROX_BOT    ;BOTAO ENTER
                JB    INS_MASSA,INC_PR  ;BOTAO INCREMENTA MASSA
                MOV   P1,POS_DISP       ;POSICAO NO DISPLAY
                LCALL INSTRUCAO         ;PASSA INTRUCAO PARA O DISPLAY
                INC   @R0               ;INCREMENTA DIGITOS DA MASSA
```

```
            MOV   A,@R0           ;JOGA DIGTOS INCREMENTADOS P ACUMULADOR
            CJNE  A,DIG_MAX,CONT_M ;SE FOR
            MOV   @R0,#$00        ;INICIALIZA O DIGITO
CONT_M:
            MOV   A,@R0
            ADD   A,#$30
            MOV   P1,A            ;VALOR DOS DIGITOS NO DISPLAY
            LCALL DADO            ;MOSTRA VALOR NO DISPLAY

DEB:
            MOV   R2,#$05         ;DEFINE O TEMPO DE 5 MILI NO DEBAUS
            LCALL DELAY           ;ROTINA PARA TEMPORIZAR 5 MILI
            JNB   INS_MASSA,DEB   ;LIBERA QUANDO O BOTAO ESTIVER SOLTO
            LJMP  INC_PR

PROX_BOT:
            RET


C_MAS:
            MOV   R2,#$05        ;DEFINE O TEMPO DE 5 MILI NO DEBAUS
            LCALL DELAY          ;ROTINA PARA TEMPORIZAR 5 MILI
            JNB   ENTER,C_MAS    ;ESPERA A TECLA SER LIBERADA
            MOV   POS_DISP,#$C1  ;POSICAO DO VALOR NO DISPLAY
            MOV   DIG_MAX,#$02   ;VALOR MAXIMO DO DIGITO
            MOV   R0,#$2A        ;POSICAO ONDE SERA ARMAZENADA A CENTENA DE
MASSA
            LCALL INSE_MASSA     ;INSERE MASSA NO DISPLAY
            RET

D_MAS:
            MOV   R2,#$05        ;DEFINE O TEMPO DE 5 MILI NO DEBAUS
            LCALL DELAY          ;ROTINA PARA TEMPORIZAR 5 MILI
            JNB   ENTER,D_MAS    ;ESPERA A TECLA SER LIBERADA

            MOV   POS_DISP,#$C2  ;POSICAO DO VALOR NO DISPLAY
            MOV   DIG_MAX,#$0A   ;VALOR MAXIMO DO DIGITO
            MOV   R0,#$2B        ;POSICAO ONDE SERA ARMAZENADA DEZENA DE
MASSA
            LCALL INSE_MASSA     ;INSERE MASSA NO DISPLAY
            RET

U_MAS:
            MOV   R2,#$05        ;DEFINE O TEMPO DE 5 MILI NO DEBAUS
            LCALL DELAY          ;ROTINA PARA TEMPORIZAR 5 MILI
            JNB   ENTER,U_MAS    ;ESPERA A TECLA SER LIBERADA

            MOV   POS_DISP,#$C3  ;POSICAO DO VALOR NO DISPLAY
            MOV   DIG_MAX,#$0A   ;VALOR MAXIMO DO DIGITO
            MOV   R0,#$2C        ;POSICAO ONDE SERA ARMAZENADA UNIDADE DE
MASSA
            LCALL INSE_MASSA     ;INSERE MASSA NO DISPLAY
            RET

;#####################ROTINAS DE DISTANCIA#####################################
```

```
DIST_PERCORRIDA:
             INC   @R0               ;CONTA DE UM EM UM METRO
S_NOVE:
             CJNE  @R0,#10,SAI       ;VERIFICA O FIM DA CONTAGEM
             MOV   @R0,#00           ;ZERA A UNIDADE
             DEC   R0                ;PROXIMO PONTEIRO DA RAM
             INC   @R0               ;CONTA A DEZENA
             MOV   A,R0
             CJNE  A,#IN_MEMO,S_NOVE ;TESTA SE CHEGOU NO INICIO DA MEMO
             CJNE  @R0,#10,SAI       ;VERIFICA O FIM DA CONTAGEM
             MOV   @R0,#00           ;ZERA A DEZENA
SAI:
             RET                     ;FIM DE ROTINA

;######################ROTINAS DE VELOCIDADE####################################

CONV_VEL:

;A= INTEIRO DE (DIST*6/10)
;B= RESTO DE (DIST*6/10)
;INTEIRO=A*6
;A = B*6
;VELOCIDADE = A/10 + INTEIRO

             MOV   A,R3
             MOV   B,#06
             MUL   AB
             MOV   B,#10
             DIV   AB
             MOV   RESTO,B
             MOV   B,#6
             MUL   AB
             MOV   INTEIRO,A
             MOV   A,RESTO
             MOV   B,#06
             MUL   AB
             MOV   B,#10
             DIV   AB
             ADD   A,INTEIRO
             RET

DEC_BCD_3D_VEL:

             MOV   B,#10          ;DIVISAO POR 10 PARA CONVERTER PARA BCD
             DIV   AB             ;CONVERTE PARA BCD
             MOV   U_VEL,B        ;UNIDADE DE VELOCIDADE
             MOV   B,#10          ;DIVISAO POR 10 PARA CONVERTER PARA BCD;
             DIV   AB             ;CONVERTE PARA BCD
             MOV   D_VEL,B        ;DEZENA DE VELOCIDADE
             MOV   C_VEL,A        ;CENTENA DE VELOCIDADE
             RET

EXIBIR_VEL:

             LCALL CONV_VEL       ;ROTINA DE CONVERSAO DE VELOCIDADE
             LCALL DEC_BCD_3D_VEL ;ROTINA DE CONVERSAO DE DEC PARA BCD
             MOV   P1,#$C0        ;POSICAO INICIAL DA VELOCIDE NO DISPLAY
```

```
            LCALL INSTRUCAO      ;ROTINA QUE PASSA INSTRUCAO PARA O DISPLAY
            MOV   R0,#$27        ;ENDERECO INICIAL DA VELOCIDADE
            MOV   FIM_EXIB,#$2A  ;ULTIMO ENDERECO A NAO SER EXIBIDO
            LCALL EXIBIR_DISP    ;ROTINA PARA EXIBIR A MENSAGEM NO DISPLAY
            RET

;######################ROTINAS DE CALORIA#####################################

CONVERT_MASSA:

            MOV  A,C_MASSA       ;CENTENA DA CALORIA
            MOV  B,#100          ;
            MUL  AB              ;MULTIPLICA P CONCATENAR COM OUTRAS UNIDADES
            MOV  R4,A            ;SALVA O VALOR EM R4
            MOV  A,D_MASSA       ;DEZENA DA CALORIA
            MOV  B,#10
            MUL  AB              ;MULTIPLICA P CONCATENAR COM OUTRAS UNIDADES
            ADD  A,R4            ;CONCATENOU DEZENA COM CENTENA
            ADD  A,U_MASSA       ;CONCATENOU UNIDADE, DEZENA E CENTENA
            MOV  R4,A            ;R4 TEM AGORA O MASSA DA PESSOA
            RET

CONVERT_CAL:
            LCALL CONVERT_MASSA  ;ROTINA CONCATENA 3 DIGITOS PARA UM
REGISTRADADO
            MOV   A,R4           ;VALOR DO MASSA CONCATENADO
            MOV   B,#15          ;COEFICIENTE DE CALCULO DA CALORIA
            DIV   AB             ;CALCULA A CALORIA
            MOV   INTEIRO_CAL,A  ;SALVA O INTEIRO DA CALORIA
            MOV   CALORIAS,A
            MOV   A,B            ;RESTO DA DIVISAO
            MOV   B,#10
            MUL   AB
            MOV   B,#15
            DIV   AB
            MOV   RESTO_CAL,A    ;SALVA O RESTO DA CALORIA
            CLR   C
            SUBB  A,#05          ;SUBTRAI O RESTO POR 5
            JC    N_ARREDONDA    ;SE NEGATIVO, SALTA
            INC   CALORIAS       ;SE POSITIVO, INCREMENTA CALORIA
N_ARREDONDA:
            RET

BCD_MENOS_SIG:

            MOV   B,#10          ;VALOR PARA CONVERTER PARA BCD
            DIV   AB             ;CONVETIDO PARA BCD
            MOV   U_CAL,B        ;UNIDADE DA CALORIA
            MOV   D_CAL,A        ;DEZENA DA CALORIA
            RET

BCD_MAIS_SIG:

            MOV   B,#10          ;VALOR PARA CONVERTER PARA BCD
            DIV   AB             ;CONVETIDO PARA BCD
            MOV   C_CAL,B        ;CENTENA DA CALORIA
            MOV   M_CAL,A        ;MILHAR DA CALORIA
```

```
            RET

INT_CAL:

            MOV     A,INTEIRO_CAL  ;INTEIRO DA CALORIA
            MOV     B,R5           ;QUANTIDADE DE 30S
            MUL     AB             ;
            MOV     MAIS_SIG,B     ;PARTE MAIS SIG DO CALCULO
            MOV     MENOS_SIG,A    ;PARTE MENOS SIG DO CALCULO
            RET

REST_CAL:
            MOV     A,RESTO_CAL    ;RESTO DA CALORIA
            MOV     B,R5           ;QUANTIDADE DE 30S
            MUL     AB
            MOV     CAL_MAIS_SIG,B ;PARTE MAIS SIG DO CALCULO
            MOV     CAL_MENOS_SIG,A;PARTE MENOS SIG DO CALCULO
            RET

DIVISAO_P_10:

            MOV     B,#00          ;PARTE INTEIRA DA DIVISAO
            MOV     A,CAL_MENOS_SIG;
DIV_N:
            CLR     C
            SUBB    A,#10          ;DIVISOR
            JNC     N_NEGA         ;VERIFICAR SE O VALOR E NEGATIVO
            DEC     CAL_MAIS_SIG   ;SUBTRAI UM NO CAL_MAIS SIG
N_NEGA:
            INC     B              ;PARTE INTEIRA DA DIVISAO
            MOV     R0,#$42        ;ENDERECO DO CAL_MAIS_SIG
            CJNE    @R0,#$FF,DIV_N
            ADD     A,#10          ;DIVISOR
            MOV     CAL_MENOS_SIG,A;RESULTADO MENOS SIG DA DIVISAO
            DEC     B              ;
            MOV     CAL_MAIS_SIG,B ;RESULTADO MAIS SIG DA DIVISAO
            RET

CAL_F_TEMPO:
            CALL    INT_CAL        ;MULTIPLICA INTEIRO DA CALORIA PELA
QUANTIDADE DE 30S
            CALL    REST_CAL       ;MULTIPLICA RESTO DA CALORIA PELA
QUANTIDADE DE 30S
            CALL    DIVISAO_P_10   ;ROTINA QUE FAZ A DIVISAO POR 10
            MOV     A,CAL_MENOS_SIG
            CLR     C
            SUBB    A,#05          ;SUBTRAI O MENOS SIG DO RESTO POR 5
            JC      N_ARRED        ;SE FOR NEGATIVO, SALTA
            INC     CAL_MAIS_SIG   ;SE FOR POSITIVO, INCREMENTA O MAIS SIG DO
RESTO
N_ARRED:
            MOV     A,MENOS_SIG
            ADD     A,CAL_MAIS_SIG ;SOMA O CAL_MAIS_SIG COM O MENOS_SIG
INTEIRO
            JNC     N_SOMA
            INC     MAIS_SIG
N_SOMA:
```

```
            MOV    MENOS_SIG,A
            RET


DIVISAO_P_100:

            MOV    B,#00          ;PARTE INTEIRA DA DIVISAO
            MOV    A,MENOS_SIG    ;
DIV_NOV:
            CLR    C
            SUBB   A,#100         ;DIVISOR
            JNC    N_NEG          ;VERIFICAR SE O VALOR E NEGATIVO
            DEC    MAIS_SIG       ;SUBTRAI UM NO MAIS SIG
N_NEG:
            INC    B              ;PARTE INTEIRA DA DIVISAO
            MOV    R0,#$39        ;ENDERECO DO MAIS_SIG
            CJNE   @R0,#$FF,DIV_NOV;VERIFICA SE O CONTEUDO E FF
            ADD    A,#100         ;DIVISOR
            MOV    MENOS_SIG,A    ;RESULTADO MENOS SIG DA DIVISAO
            DEC    B              ;
            MOV    MAIS_SIG,B     ;RESULTADO MAIS SIG DA DIVISAO
            RET

DEC_BCD_3D_CAL:

            MOV   B,#10          ;DIVISAO POR 10 PARA CONVERTER PARA BCD
            DIV   AB             ;CONVERTE PARA BCD
            MOV   U_CAL,B        ;UNIDADE DE VELOCIDADE
            MOV   B,#10          ;DIVISAO POR 10 PARA CONVERTER PARA BCD;
            DIV   AB             ;CONVERTE PARA BCD
            MOV   D_CAL,B        ;DEZENA DE VELOCIDADE
            MOV   C_CAL,A        ;CENTENA DE VELOCIDADE
            RET

EXIBIR_CAL:

            CJNE  R5,#$01,V_MAIOR;APOS 30 SEGUNDO CONVERTE EM CALORIA
            LCALL CONVERT_CAL    ;CONVERTE MASSA EM CALORIA
            MOV   A,CALORIAS     ;
            LCALL BCD_MENOS_SIG  ;CONVERTE O VALOR DA CALORIA PARA BCD
            MOV   P1,#$CE        ;POSICAO DO DISPLAY ONDE VAI FICAR A D_CAL
            CALL  INSTRUCAO      ;ROTINA QUE ENVIA INSTRUCAO PARA O DISPLAY
            MOV   R0,#$2F        ;ENDERECO DA D_CAL
            MOV   FIM_EXIB,#$31  ;ULTIMO ENDERECO PARA NAO SER EXIBIDO
            LCALL EXIBIR_DISP    ;EXIBI O VALOR NO DISPLAY
            RET

V_MAIOR:

            LCALL CAL_F_TEMPO    ;ROTINA QUE MULTIPLICA A CALORIA
            LCALL DIVISAO_P_100  ;ROTINA QUE FAZ DIVISAO POR 100
            MOV   A,MAIS_SIG     ;VALOR MAIS SIGNIFICATIVO
            JNZ   QUATRO_D       ;TESTA SE O MAIS SIGNIFICATIVO E ZERO
            MOV   A,MENOS_SIG    ;
            LCALL DEC_BCD_3D_CAL ;CONVERTE PARA TRES DIGITOS
            MOV   P1,#$CD        ;POSICAO NO DISPLAY ONDE VAI FICAR A DEZENA
            LCALL INSTRUCAO      ;ROTINA QUE PASSA INSTRUCAO PARA O DISPLAY
```

```
            MOV   R0,#$2E        ;ENDERECO ONDE ESTAR A CENTENA DA CAL
            MOV   FIM_EXIB,#$31  ;
            LCALL EXIBIR_DISP    ;EXIBI VALORES NO DISPLAY
            RET
QUATRO_D:

            MOV   A,MENOS_SIG    ;
            LCALL BCD_MENOS_SIG  ;CONVERTE VALOR MENOS SIG EM BCD
            MOV   A,MAIS_SIG
            LCALL BCD_MAIS_SIG   ;CONVERTE VALOR MAIS SIG EM BCD
            MOV   P1,#$CC        ;POSICAO DO DISPLAY QUE VAI FICAR A CENTENA
            LCALL INSTRUCAO      ;ROTINA QUE PASSA INSTRUCAO PARA O DISPLAY
            MOV   R0,#$2D        ;ENDERECO ONDE ESTAR O MILHAR DA CAL
            MOV   FIM_EXIB,#$31  ;ENDERECO DO MILHAR DA CALORIA
            LCALL EXIBIR_DISP    ;EXIBI VALORES NO DISPLAY
            RET


;######################ROTINAS DE LIMPAR MEMORIA##############################

MEMO_ZERADA:
            MOV   R0,#$21        ;ENDERECO INICIAL DA MEMORIA RAM
ZERA_M:
            MOV   @R0,#$00       ;CONTEUDO DA MEMORIA ZERADO
            INC   R0             ;PROXIMO ENDERECO
            CJNE  R0,#$39,ZERA_M ;TESTA SE E A ULTMA POSICAO ZERADA
            RET

;######################KERNEL PRINCIPAL#######################################

INICIO:
            LCALL INICIADISP     ;INICIALIZACAO DO DISPLAY
            LCALL MENS_INICIAL   ;ROTINA DE MENSAGEM DE SAUDACAO
            MOV   P1,#$01        ;INSTRUCAO PARA LIMPAR O DISPLAY
            LCALL INSTRUCAO      ;ROTINA QUE PASSA A INSTRUCAO PARA O DISPLAY
            LCALL MENS_MASSA     ;ROTINA QUE RETORNA MENSAGEM DE MASSA
            LCALL MEMO_ZERADA    ;ROTINA PARA LIMPAR MEMORIA

            JB    ENTER,$        ;BOTAO ENTER
            LCALL C_MAS          ;INS NO DISP E SALVA NA MEMO CENT DA MASSA
            LCALL D_MAS          ;INS NO DISP E SALVA NA MEMO DEZ DA MASSA
            LCALL U_MAS          ;INS NO DISP E SALVA NA MEMO UNID DA MASSA
            LCALL MENS_UNIDADE   ;MEN KM/H, METROS E KCAL

            MOV   R3,#$FF        ;INICIO DA DISTANCIA
            MOV   R6,#25         ;TEMPO ATE 30 SEGUNDOS
            MOV   R5,#00         ;MULT O VALOR DA CALORIA ENCONTRADO
            MOV   R0,#$26        ;INICIALIZA A UNIDADE DE DISTANCIA
            MOV   @R0,#$FF       ;INICIALIZADA
            MOV   SEGUNDO,#16    ;VALOR QUE MUL COM 62500 TOTALIZA 1 SEG
            MOV   N_ASCIO,#03    ;TEMPO PARA DESLIGAR O TIMER
            CLR    TR1           ;DESLIGA O TIMER 1
            LCALL TEMP           ;INICIALIZA TIMER 1

BEGIN:

            JNB   SENSOR,VAL_DIST;SENSOR DE VOLTAS
```

```
            JNB   TF1,BEGIN      ;VERIFICAR SE OS 62500 CHEGOU NO FIM
            LCALL TEMP           ;INICIALIZA TIMER 1
            DJNZ  SEGUNDO,BEGIN  ;VERIFICAR SE PASSOU 1 SEGUNDO
            LCALL EXIBIR_VEL     ;ROTINA DE EXIBICAO DE VELOCIDADE
            MOV   R3,#$00

            DJNZ  N_ASCIO,SEG_N  ;SE O SENSOR NAO FOR ASCIONADO EM 3 SEG
            CLR   TR1            ;DESLIGA O TIME
            MOV   R6,#25         ;INICIALIZA OS 30 SEG
            MOV   R3,#$FF        ;INICIO DA DISTANCIA
SEG_N:
            MOV   SEGUNDO,#16    ;VALOR QUE MUL COM 62500 TOTALIZA 1 SEG
            DJNZ  R6,BEGIN       ;VERIFICAR SE CHEGOU EM 30 SEGUNDO
            INC   R5             ;VALOR PARA MULTIPLICACAO DA CALORIA
            LCALL EXIBIR_CAL     ;ROTINA QUE EXIBE A CALORIA
            MOV   R6,#25         ;INICIALIZA OS 30 SEG
            LJMP  BEGIN

VAL_DIST:
            MOV   N_ASCIO,#03    ;TEMPO PARA DESLIGAR O TIMER
            SETB  TR1            ;LIGA O TIMER 1
            MOV   R2,#10         ;DEFINE O TEMPO DE 5 MILI NO DEBAUS
            LCALL DELAY          ;ROTINA PARA TEMPORIZAR 5 MILI

            JNB   SENSOR,$       ;ESPERA A TECLA SER LIBERADA
            INC   R3             ;VALOR DA DIST PECOR NO INSTANTE
            MOV   R0,#$26        ;UNIDADE DA DISTANCIA
            MOV   IN_MEMO,#$21   ;E PASSADO O INICIO DA MEMORIA PRA ROTINA
            LCALL DIST_PERCORRIDA ;VALOR DA DISTANCIA PECORRIDA
            MOV   P1,#$C5        ;ENDERECO INICIAL DA DISTANCIA NO DISPLAY
            LCALL INSTRUCAO      ;ROTINA QUE PASSA INSTRUCAO PARA O DISPLAY
            MOV   R0,#$21        ;VALOR DA CENTENA DE MILHAR A SER EXIBIDO
            MOV   FIM_EXIB,#$27  ;ENDERECO A NAO SER EXIBIDO PELA ROTINA
            LCALL EXIBIR_DISP    ;ROTINA DE EXIBICAO DA DISTANCIA NO DISPLAY
            LJMP  BEGIN
            END                  ;FIM DA ROTINA
```