**CEFET-SP** **Microcontroladores – Família MCS-51**

**Conceitos, Aplicações e Projetos** **versão 6.0**

**2004** **Wilson Ruiz**

***Harpia harpyja*** **Harpia ou Águia Real** **Brasil**

# CAPÍTULO 1: HARDWARE DOS MICROCONTROLADORES DA FAMÍLIA INTEL 8051 (MCS-51)

## 1.1 CARACTERÍSTICAS PRINCIPAIS COMUNS AOS MICROCONTROLADORES DA FAMÍLA INTEL MCS-51:

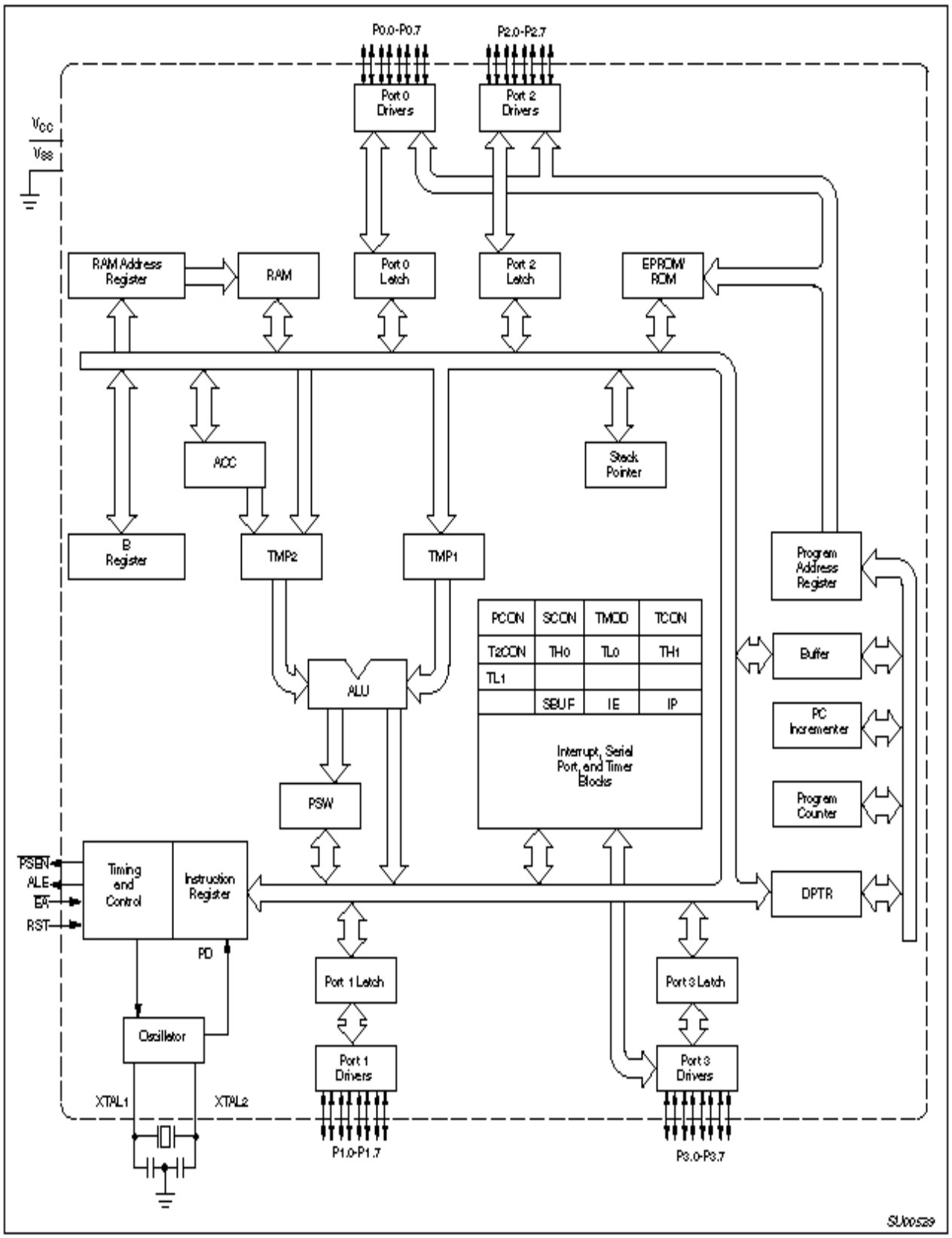
- **Família de microcontroladores mais usada atualmente.**
- **CPU de 8 bits otimizada para aplicações de controle.**
- **Clock típico de 12MHz (valor usado em aplicações gerais existindo também versões mais rápidas).**
- **Capacidade de 64 Kbytes de memória de programa (ROM) e 64 Kbytes de memória de dados (RAM).**
- **4 Kbytes de memória de programa interna (ROM interna).**
- **RAM interna com 128 bytes (há versões com capacidades superiores).**
- **4 portas de I/O de 8 bits cada, com bits individualmente endereçáveis.**
- **Interrupções mascaráveis em dois níveis de prioridades (três internas e duas externas).**
- **2 temporizadores / contadores internos de 16 bits programáveis.**
- **Oscilador de clock interno.**
- **Canal de comunicação serial.**
- **Capacidade de execução de complexas operações aritméticas e lógicas (multiplicação, divisão, permuta e deslocamento de bits etc).**
- **Família com grande variedade de CPU's, com versões diferenciando-se na especialização, porém apresentando mesma arquitetura interna básica.**
- **Fornecido por diferentes fabricantes que personalizam o seu produto mantendo a compatibilidade com as versões originais.**

## 1.2 QUADRO COMPARATIVO ENTRE ALGUMAS VERSÕES DA FAMÍLIA 8051 E OUTRAS SIMILARES DA INTEL:

| Código do dispositivo | Versão sem ROM | Versão com EPROM | RAM Interna bytes | ROM Interna bytes | Portas E/S de 8 bits | Linhas de I/O | Timers /Contadores | UART | Canais de DMA | Modos de baixo consumo e Idle |
|---|---|---|---|---|---|---|---|---|---|---|
| 8048AH | 8035AHL | 8748H | 64 | 1 K | | 27 | 1 | - | - | - |
| 8049AH | 8039AHL | 8749H | 128 | 2 K | | 27 | 1 | - | - | - |
| 8050AH | 8040AHL | - | 256 | 4 K | | 27 | 1 | - | - | - |
| 8051 | 8031 | - | 128 | 4 K | 4 | 32 | 2 | sim | - | - |
| 8051AH | 8031AH | 8751AH | 128 | 4 K | 4 | 32 | 2 | sim | - | - |
| 80C51BH | 80C31BH | 87C51 | 128 | 4 K | 4 | 32 | 2 | sim | - | sim |
| 8052AH | 8032AH | 8752BH | 256 | 8 K | 4 | 32 | 3 | sim | - | sim |
| 80C52 | 80C32 | - | 256 | 8K | 4 | 32 | 3 | sim | - | sim |
| 83C51FA | 80C51FA | 87C51FA | 256 | 8K | 4 | 32 | 3 | sim | - | sim |
| 83C51FB | 80C51FB | 87C51FB | 256 | 16K | 4 | 32 | 3 | sim | - | sim |
| 83C152JA | 80C152JA | - | 256 | 8K | 5 | 40 | 2 | sim | 2 | sim |
| - | 80C152JB | - | 256 | - | 7 | 56 | 2 | sim | 2 | sim |
| 83C152JC | 80C152JC | - | 256 | 8K | 5 | 40 | 2 | sim | 2 | sim |
| - | 80C152JD | - | 256 | - | 7 | 56 | 2 | sim | 2 | sim |
| 83C452 | 80C452 | 87C452P | 256 | 8K | 5 | 40 | 2 | Sim | - | sim |

**OBSERVAÇÕES:**

- **A letra “C” usada em alguns códigos indica que a tecnologia usada no projeto é a CHMOS, que apresenta um menor consumo de potência que o da tecnologia HMOS (dispositivos sem letra nos códigos).**
- **Dispositivos HMOS e CHMOS são intercambiáveis.**
- **O par de letras “BH” refere-se ao projeto interno do chip.**
- **Os modos “Idle” (preguiçoso, ocioso) e “de baixa potência” são programados por software e tem o objetivo de reduzir o consumo, neles a CPU é desligada, enquanto a RAM e outros periféricos internos continuam a operar.**




## 1.3 ARQUITETURA INTERNA DA FAMÍLIA 8051

# 1.4 DESCRIÇÃO DA PINAGEM DO 8051

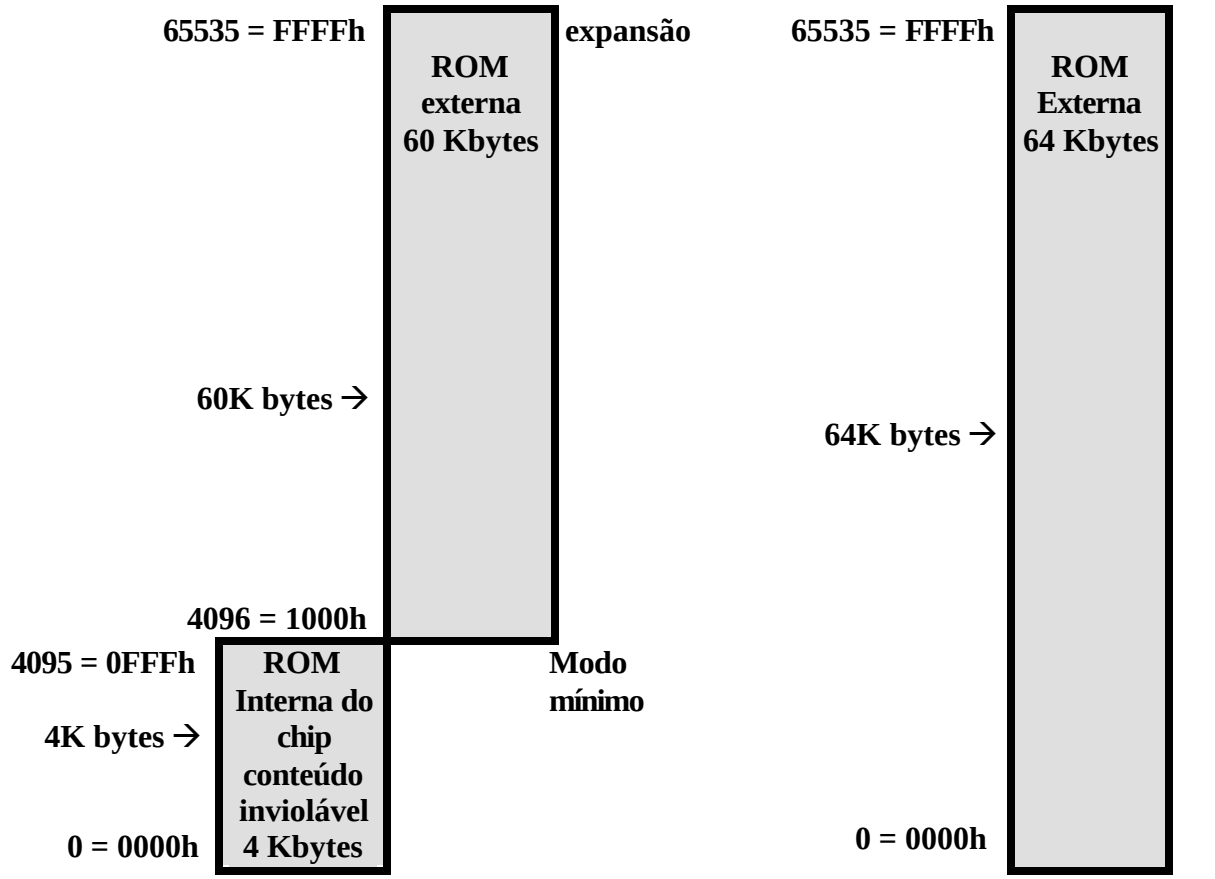
| Número dos Pinos | Nome | Descrição resumida de sua função |
|---|---|---|
| 1 a 8 | P1.0 a P1.7 Port 1 | Porta de I/O número 1 |
| 9 | RST/ VPD | "Reset" do sistema (é necessário a aplicação de um nível alto TTL, durante 2 ou mais ciclos de máquina) |
| 10 a 17 | P3.0 a P3.7 Port 3 | Porta de I/O número 3<br>Possibilita também funções especiais relacionadas ao Canal Serial, Interrupções e "Timer/Counter" |
| 18 | XTAL 2 | Ligação do cristal oscilador |
| 19 | XTAL 1 | Ligação do cristal oscilador |
| 20 | Vss | Terra |
| 21 a 28 | P2.0 a P2.7 Port 2 ou A8 a A15 | Porta de I/O número 2<br>Saída do byte mais significativo do endereço, para memória externa. |
| 29 | PSEN' | Program Store Enable<br>"Strobe" da memória de programa externa. Quando o sistema lê instruções ou operandos na memória externa, vai para nível zero e não é ativado (nível 1) durante busca na memória interna de programa. |
| 30 | ALE / PROG | Address Latch Enable<br>Saída para habilitar o "latch" de endereços. Serve para separar, a parte menos significativa do endereço, dos dados, na aplicação de memória externa.<br>Entrada do pulso de programação durante a gravação da EPROM. |
| 31 | EA' / Vpp | External Access Enable – Programming Supply Voltage<br>Entrada de seleção da memória de programa. Quando em nível zero, a CPU trabalha apenas com a memória de programa externa.. Se em nível lógico 1, a CPU executa instruções da memória de programa interna, desde que o PC seja menor que 4096. Este pino recebe +21 volts durante a programação da ROM interna.<br>Recebe +21V durante a programação da EPROM |
| 32 a 39 | P0.0 a P0.7 ou AD0 a AD7 | Porta de I/O número 0<br>Fornece o byte menos significativo de endereço multiplexado com os dados. |
| 40 | Vcc | + 5 volts |

# FUNÇÕES ESPECIAIS DOS PINOS DA PORTA 3

| Nomes do Pino | Descrição resumida de sua função |
|---|---|
| **P3.0 = RXD/data** | **Receptor da porta serial assíncrona ou entrada e saída de dados síncronos (expansão de I/O pela porta serial).** |
| **P3.1 = TXD/ clock** | **Saída de transmissão da porta serial assíncrona, ou saída de clock para os registradores de deslocamento externos (expansão de I/O pela porta serial).** |
| **P3.2 = INT0'** | **Interrupção externa número 0, ou<br>Bit de controle para o "timer/counter" 0.** |
| **P3.3 = INT1'** | **Interrupção externa número 1, ou<br>Bit de controle para o "timer/counter" 1.** |
| **P3.4 = T0** | **Entrada externa para o "timer/counter" 0.** |
| **P3.5 = T1** | **Entrada externa para o "timer/counter" 1.** |
| **P3.6 = WR'** | **"Strobe" (sincronismo) de escrita na memória de dados externa (escrita na memória RAM).** |
| **P3.7 = RD'** | **Strobe (sincronismo) de leitura da memória de dados externa<br>(leitura da memória RAM).** |

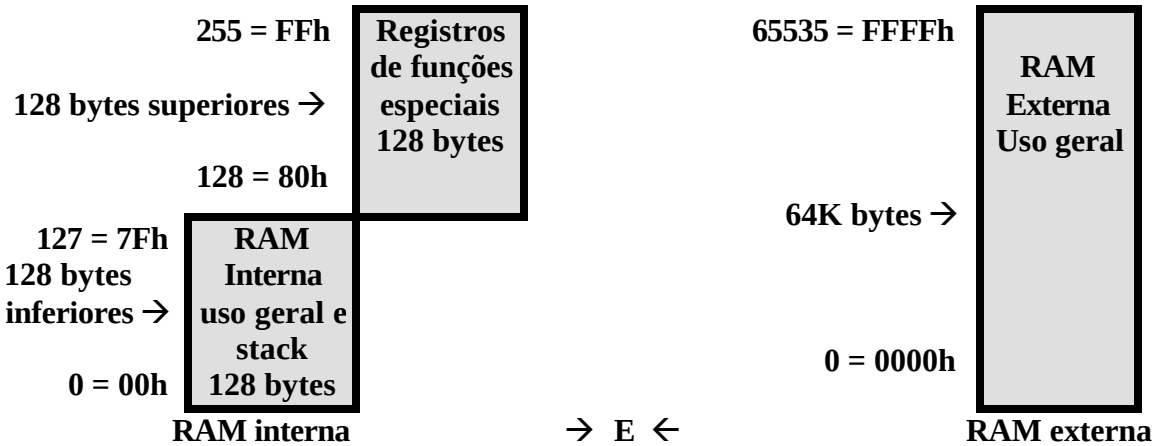
# 1.5 ORGANIZAÇÃO DA MEMÓRIA DA FAMÍLIA 8051 (e equivalentes)

**MAPA DA MEMÓRIA DE PROGRAMA (ROM)**




**Para o 8051: pino EA'=1** → **OU** ← **Para o 8031: pino EA' = 0**

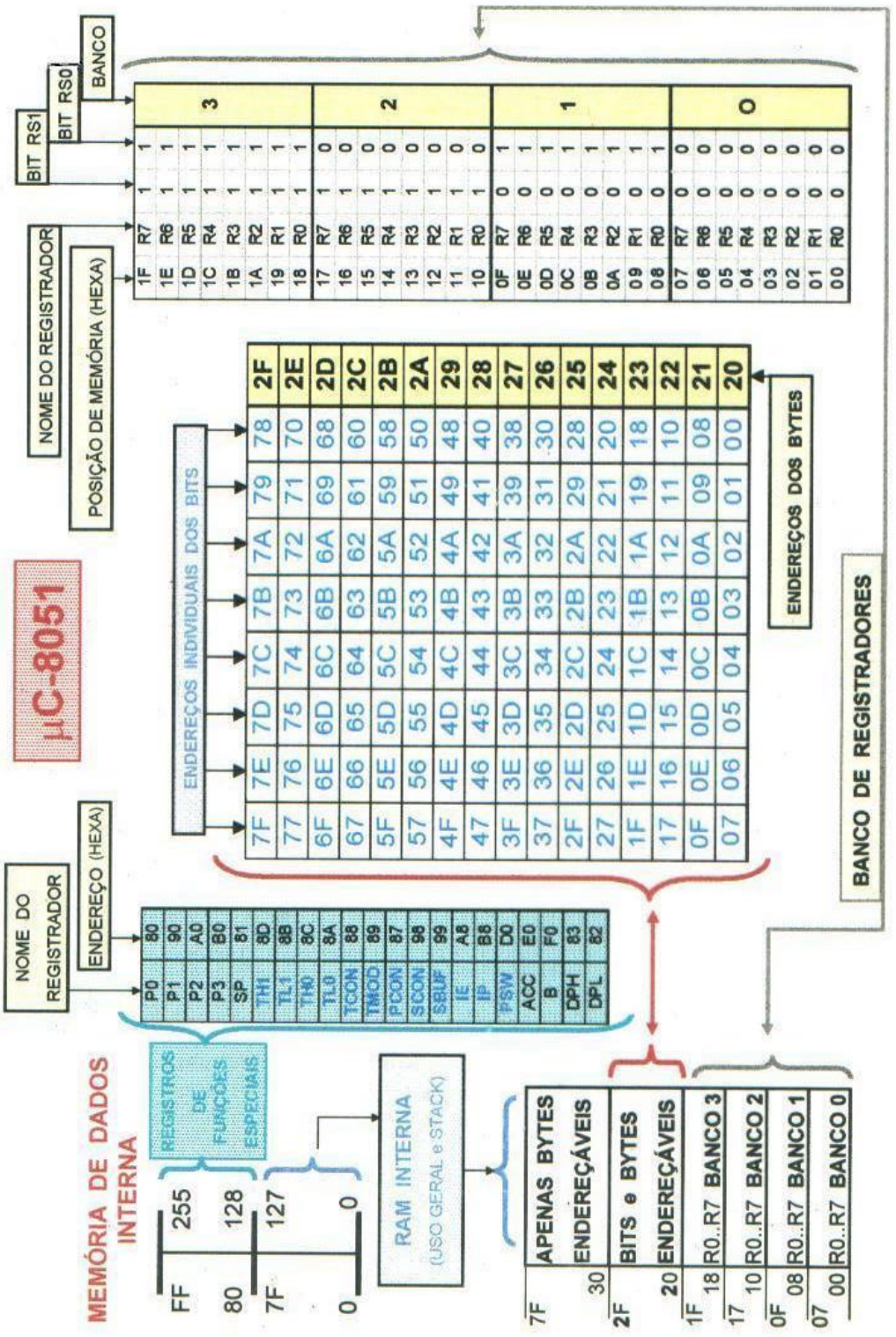
**MAPA DA MEMÓRIA DE DADOS (RAM)**



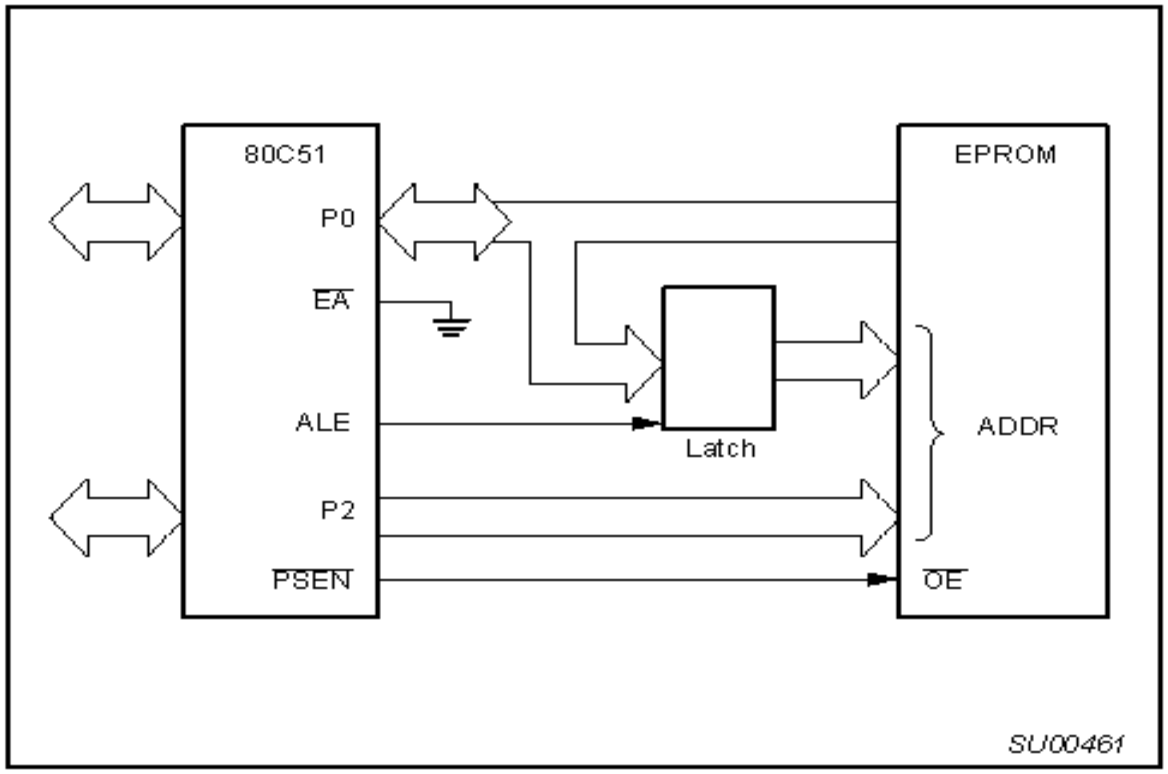

**RAM interna** → **E** ← **RAM externa**

******* A RAM externa não é endereçada diretamente pela CPU. O endereço de acesso deve estar em um registrador usado pela CPU como índice.**

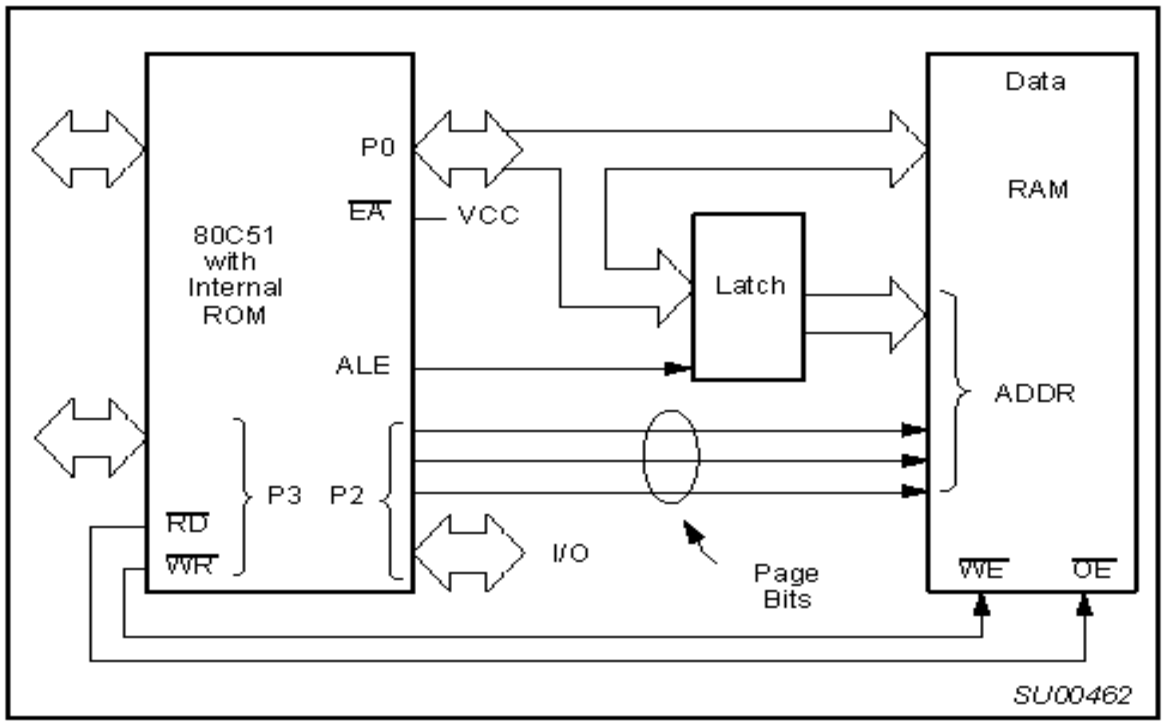
# ESTRUTURA DA MEMÓRIA RAM INTERNA



## 1.6 LIGAÇÃO COM A MEMÓRIA EXTERNA



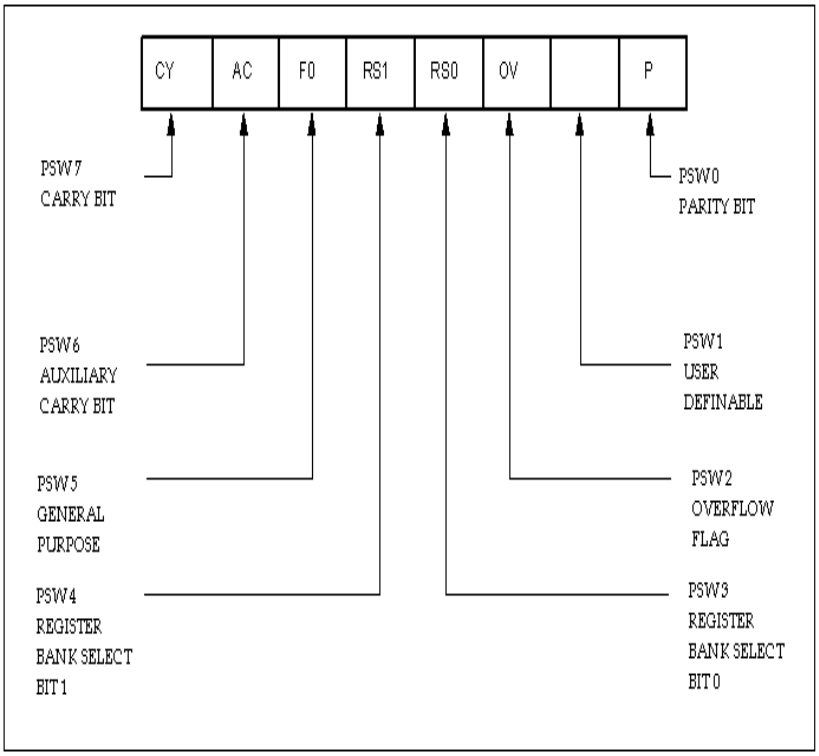

EXECUÇÃO A PARTIR DE MEMÓRIA EXTERNA




ACESSO À MEMÓRIA DE DADOS EXTERNA

## 1.7 PROGRAM STATUS WORD (PSW) - endereço D0h

**Este byte, localizado no espaço SFR (Special Status Word – RAM interna de 80h a FFh), possui alguns bits de status que refletem o estado da CPU. Mostrado na figura abaixo, contém as flags: Carry, Auxiliay Carry, Overflow, Paridade, dois bits (RS1 e RS0) de seleção de banco de registradores e dois bits de status definidos pelo usuário.**




**Registrador PSW ( Program Status Word )**

| Flag | | Descrição | |
|---|---|---|---|
| C = PSW.7 | Flag carry | C = 1 indica o transporte no bit 7 (vai 1 ou vem 1), C = 0 caso contrário | |
| AC = PSW.6 | Flag auxiliar carry | C = 1 indica o transporte entre os bits 3 e 4 (vai 1 ou vem 1), C = 0 caso contr. | |
| F0 = PSW.5 | Flag de uso geral | Pode-se setar ou resetar esta flag, indicando o status da condição escolhida | |
| RS1 = PSW.4<br>RS0 = PSW.3 | Seleção do banco de registradores | RS1 RS0 = 00 → banco 0 | RS1 RS0 = 10 → banco 2 |
| | | RS1 RS0 = 01 → banco 1 | RS1 RS0 = 11 → banco 3 |
| OV = PSW.2 | Flag de overflow | OV = 1 indica uma condição de erro, o resultado não pode ser representado como um nº sinalizado (ex. soma de nºs negativos resultando em positivo) | |
| P = PSW.0 | Flag de paridade | P = 0 p/ paridade par do acumulador e P = 1 p/ paridade impar do acumulador | |

## 1.8 RESET – principais características:

- **Ativo quando o pino RST (9) permanecer em "nível 1" por 2 ou mais ciclos de máquina.**

- **RESET interno:**
  - **PC, Acumulador, B, Flags, DPTR, registros dos temporizadores / contadores são zerados.**
  - **SP ← 07h**
  - **SBUF (buffer serial) estará com conteúdo indeterminado e o registro de controle da porta serial (SCON) será zerado.**
  - **PCON terá apenas o seu bit +significativo zerado.**
  - **IE e IP (registros de controle de interrupção) terão xxx00000.**
  - **As portas P0 a P3 terão o valor FFh. (durante o RESET, o nível nos pinos é indeterminado, indo a "nível 1" após a execução da rotina interna de RESET assim, o hardware externo, dependendo da aplicação, deve prever essa situação, evitando o acionamento indesejado de algum periférico.**
  - **A RAM interna não é afetada pelo RESET "forçado", sendo que após o "power-up" seu valor é aleatório.**

## 1.9 INTERRUPÇÃO

**Existem três fontes de interrupção:**

- **A interrupção por software (instrução)**
- **A interrupção solicitada por um periférico externo**
- **A interrupção solicitada por um periférico interno (Timer/Counter ou Porta Serial)**

**Vantagem da interrupção:**

- **Simplificação do hardware, pois o sistema não necessita ficar monitorando o funcionamento de alguns dispositivos externos.**

**Características das Interrupções:**

- **Mascaramento:**
  - **Possibilidade do sistema atender ou não uma solicitação de interrupção (mascaráveis).**
  - **Existem sistemas com interrupções não mascaráveis.**

- **Prioridades:**
  - **Para um sistema que tenha mais de uma interrupção, existe uma tabela de prioridades, que determina qual deve ser atendida primeiramente no caso de solicitações coincidentes.**

- **Interrupção vetorada:**
  - **Possuem o "Vetor de Interrupção" (endereço de início da interrupção) fixo, não sendo definido pelo usuário ( família 8051).**

- **Interrupção não vetorada:**
  - **Os endereços de tratamento da interrupção são escolhidos pelo programador (outras famílias).**
  - **OBS: O significado das expressões "interrupção vetorada" e "interrupção não vetorada" pode ser exatamente contrário ao descrito, dependendo da convenção do fabricante da família de microcontroladores usada.**

- **Reconhecimento da interrupção:**
  - **Pelo nível (1 ou 0).**
  - **Pela borda de subida ou de descida.**
  - **Pela soma de borda (subida ou descida) e um nível correspondente.**

## AS INTERRUPÇÕES DA FAMÍLIA 8051 (e equivalentes)

**Essa família de microcontroladores apresenta 5 / 6 formas de ser interrompido:**

- **Pela interrupção externa INT0' _____(maior prioridade)**
- **Pelo timer/counter interno TIMER 0**
- **Pela interrupção externa INT1'**
- **Pelo timer/counter interno TIMER 1**
- **Pelo canal de comunicação serial_____ (menor prioridade)**
- **Pelo timer/counter interno TIMER 2 (somente para o 8032 / 52)**

**OBSERVAÇÕES:**

- **Todas as interrupções podem ser habilitadas ou não (exceto o RESET).**
- **Se o pedido da interrupção que ocorrer for de nível menor ou igual (mesmo número) ao da que já está sendo atendida, este ficará aguardando o término da mesma para ser executado.**
- **No caso de uma interrupção de prioridade menor estar em andamento, esta poderá ser interrompida por uma de prioridade maior, que ao seu término possibilita que a primeira seja concluída.**

# ENDEREÇOS DE DESVIO DAS INTERRUPÇÕES

## Mapa da memória de programa

| ENDEREÇO | NOME DA INTERRUPÇÃO | Intervalo |
|---|---|---|
| 0000h | Reset | 3 bytes |
| ... | | |
| 0002h | | |
| 0003h | INT0 | 8 bytes |
| ... | | |
| 000Ah | | |
| 000Bh | Timer/Counter 0 | 8 bytes |
| ... | | |
| 0012h | | |
| 0013h | INT1 | 8 bytes |
| ... | | |
| 001Ah | | |
| 001Bh | Timer/Counter 1 | 8 bytes |
| ... | | |
| 0022h | | |
| 0023h | Canal Serial | 8 bytes |
| ... | | |
| 002Ah | | |
| 002Bh | Timer/Counter 2<br>Somente para 8032 / 52 | 8 bytes |
| ... | | |
| 0032h | | |
| 0033h | | |

# REGISTRADORES DE CONTROLE DAS INTERRUPÇÕES

**O 8051 possui dois registradores, na Região conhecida como SFR (Registradores de Funções Especiais) para controle das interrupções.**

→ **IE (Interrupt Enable) – Endereço A8h**

**Tem por função indicar quais interrupções estão habilitadas.**

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| EA | X | X | ES | ET1 | EX1 | ET0 | EX0 |

**EA (Enable All):**

- **p/ = 0, desabilita todas as interrupções**
- **p/ = 1, permite selecionar qual interrupção será habilitada, em função dos bits de controle individuais (abaixo).**

**ES (Enable Serial):**

- **p/ = 0, inibe a interrupção solicitada pelo canal serial, independentemente do valor de EA.**
- **p/ = 1, libera a interrupção solicitada pelo canal serial, se EA = 1.**

**ET1 (Enable Timer 1):**

- **p/ = 0, inibe a interrupção solicitada pelo Timer/Counter 1, independentemente do valor de EA.**
- **p/ = 1, libera a interrupção solicitada pelo Timer/Counter 1, se EA = 1.**

**ET0 (Enable Timer 0):**

- **p/ = 0, inibe a interrupção solicitada pelo Timer/Counter 0, independentemente do valor de EA.**
- **p/ = 1, libera a interrupção solicitada pelo Timer/Counter 0, se EA = 1.**

**EX1 (Enable External 1):**

- **p/ = 0, inibe a interrupção solicitada pelo dispositivo externo ligado no pino INT1', independentemente do valor de EA.**
- **p/ = 1, libera a interrupção solicitada pelo dispositivo externo ligado no pino INT1', se EA = 1.**

**EX0 (Enable External 0):**

- **p/ = 0, inibe a interrupção solicitada pelo dispositivo externo ligado no pino INT0', independentemente do valor de EA.**
- **p/ = 1, libera a interrupção solicitada pelo dispositivo externo ligado no pino INT0', se EA = 1.**

→ **IP (Interrupt Priority) – Endereço B8h**

**Tem por função fixar qual nível de prioridade das interrupções, sendo assim possível alterar a prioridade de atendimento de uma interrupção durante o processamento.**

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| X | X | X | PS | PT1 | PX1 | PT0 | PX0 |

**PS (Priority Serial):**

- **p/ = 0, indica prioridade baixa para a interrupção solicitada pelo canal serial.**
- **p/ = 1, indica prioridade alta para esta interrupção, se a mesma estiver habilitada.**

**PT1 (Priority Timer 1):**

- **p/ = 0, indica prioridade baixa para a interrupção solicitada pelo Temporizador/Contador 1.**
- **p/ = 1, indica prioridade alta para esta interrupção, se a mesma estiver habilitada.**

**PT0 (Priority Timer 0):**

- **p/ = 0, indica prioridade baixa para a interrupção solicitada pelo Temporizador/Contador 0.**
- **p/ = 1, indica prioridade alta para esta interrupção, se a mesma estiver habilitada.**

**PX1 (Priority External 1):**

- **p/ = 0, indica prioridade baixa para a interrupção solicitada pelo dispositivo externo ligado no pino INT1'.**
- **p/ = 1, indica prioridade alta para esta interrupção, se a mesma estiver habilitada.**

**PX0 (Priority External 0):**

- **p/ = 0, indica prioridade baixa para a interrupção solicitada pelo dispositivo externo ligado no pino INT0'.**
- **p/ = 1, indica prioridade alta para esta interrupção, se a mesma estiver habilitada.**

**AJUSTE DA FORMA DE RECONHECIMENTO DAS INTERRUPÇÕES EXTERNAS**

**É possível ajustar as interrupções externas para serem detectadas pela transição de "1" para "0" ou pelo nível "0".**

→ **TCON (Timer Control) – Endereço 88h**

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| | | | | IE1 | IT1 | IE0 | IT0 |

**IT1: Bit para a escolha da forma de reconhecimento do INT1**

- **p/ = 0, indica que será aceita a interrupção apenas pelo nível "0" presente no pino INT1'.**
- **p/ = 1, indica que será aceita a interrupção na transição de "1" para "0" no nível desse pino'.**

**IT0: Bit para a escolha da forma de reconhecimento do INT0**

- **p/ = 0, indica que será aceita a interrupção apenas pelo nível "0" presente no pino INT0'.**
- **p/ = 1, indica que será aceita a interrupção na transição de "1" para "0" no nível desse pino'.**

**IE1: Bit para o hardware de controle da interrupção INT1**

- **É setado pelo hardware interno quando for detectada uma transição de "1" para "0" no pino INT1'.**
- **Tem por função sinalizar internamente o pedido da interrupção. É resetado logo que a interrupção é atendida.**

**IE0: Bit para o hardware de controle da interrupção INT0**

- **É setado pelo hardware interno quando for detectada uma transição de "1" para "0" no pino INT0'.**
- **Tem por função sinalizar internamente o pedido da interrupção. É resetado logo que a interrupção é atendida.**

## 1.10 TEMPORIZADORES E CONTADORES

**O 8051 possui 2 Temporizadores / Contadores, controláveis por programa, que podem operar de maneira totalmente independente dos demais sistemas do chip, podendo ser habilitados ou não por software ou hardware (pelos registros de controle ou pinos de interrupção).**

### → TCON (Timer Control) – Endereço 88h

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| TF1 | TR1 | TF0 | TR0 | IE1 | IT1 | IE0 | IT0 |

**OBS:**

**Os bits IE1, IT1, IE0 e ITO são usados no controle das interrupções externas, sendo referentes ao ajuste da forma de reconhecimento destas (vide item 1.8).**

**TF1:**

- **Sempre que ocorrer um overflow no T/C 1, este bit será setado, gerando um pedido de interrupção do T/C 1. É resetado pelo hardware interno ao final da rotina de interrupção.**

**TR1:**

- **Setado pelo software para ligar T/C 1. Resetado para desligar o T/C (parar contagem).**

**TF0:**

- **Sempre que ocorrer um overflow no T/C 0, este bit será setado, gerando um pedido de interrupção do T/C 0 . É resetado pelo hardware interno ao final da rotina de interrupção.**

**TR0:**

- **Setado pelo software para ligar T/C 0. Resetado para desligar o T/C (parar contagem).**

→ **TMOD (Timer Mode) – Endereço 89h**

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| Gate.1 | C/T' .1 | M 1 .1 | M 0.1 | Gate .0 | C/T'.0 | M 1.0 | M 0.0 |

**Gate.1 e Gate.0:**

- **Escolha da forma de como os T/C serão habilitados.**

**Exemplo:**

- **p/ Gate.0 = 0 →T/C 0, será habilitado (contando) fazendo-se TR0 = 1 ( no registrador TCON).**

- **p/ Gate.0 = 1 →T/C 0, será habilitado (contando) somente p/ TR0 = 1 e o pino INT 0' = 0.**

**Exemplo de aplicação:**

**Determinação da largura de um pulso, colocando-se a entrada do sinal externo no pino de interupção correspondente e fazendo com que T/C funcione com sinal interno. Desta maneira somente haverá contagem quando o sinal externo for = 1. O software calculará o tempo decorrido entre dois pulsos do sinal aplicado.**

**C/T'.1 e C/T'.0 :**

- **Selecionam a função de cada T/C individualmente**

| C/T'.0 | Função escolhida para Timer / Counter 0 |
|---|---|
| = 1 | Contagem (para sinal externo) |
| = 0 | Temporização (sinal será interno) |

| C/T'.1 | Função escolhida para Timer / Counter 1 |
|---|---|
| = 1 | Contagem (para sinal externo) |
| = 0 | Temporização (sinal será interno) |

**OBS:**

**Para qualquer um dos T/C, se o sinal for interno (timer), a freqüência será a de clock dividida por 12 e o pino de entrada correspondente fica disponível.**

**M1.1 e M0.1 :**
**M1.0 e M0.0 :**

- **Escolha de um dos possíveis modos de operação para cada T/C.**

## MODOS DE OPERAÇÃO DOS T/C's.

### MODO 0

- **Contador ou Temporizador de 8 bits, com divisor de freqüência de até 32 ($freq_{clock}$ / 32).**
- **Selecionado fazendo-se: M1.x = 0 e M0.x = 0**
- **Os registradores TL0 e TL1 servem como divisores de 5 bits (até 32).**

**TH0 ou TH1 | TL0 ou TL1**

**Contagem (bits 15...8) | ignorar (bits 7...5) Prescaler (bits 4...0)**

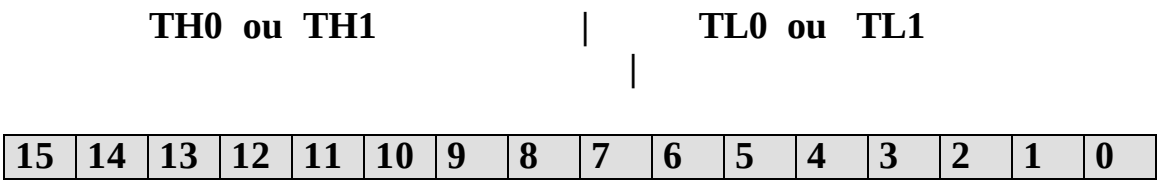
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

- **Registrador TH0 (endereço 8Ch) ou registrador TH1 (endereço 8Dh) recebe o valor inicial da contagem (valor escrito pelo software, até FFh).**
- **Ao ocorre um overflow nesse registrador, o T/C em uso gera um pedido de interrupção interno, que poderá ser ou não aceito pela CPU.**
- **O valor presente nesse registrador pode ser lido pelo software a qualquer momento.**
- **O sinal de contagem (interno ou externo) será dividido pelo valor binário presente nos bits "0" a "4" do registrador TL0 (endereço 8Ah) ou TL1 (endereço 8Bh), no T/C usado.**
- **Os bits "5"a "7" devem ser ignorados em caso de leitura.**

**Exemplo:**
**É desejada uma contagem com as seguintes características:**
**De 9Bh até FFh e que a cada 20 pulsos aplicados na entrada externa do T/C 0 (pino T0), este incremente seu registro de uma unidade.**
**Assim temos: 9Bh – FFh = 64h = 100 contagens em decimal, sendo desejado então que a cada 100 contagens internas uma interrupção seja gerada e, possibilitar assim que um determinado controle do sistema seja feito pela CPU. Dessa forma temos uma interrupção a cada 2000 sinais externos.**
**Isso é conseguido programando-se T/C 0 como contador no modo 0, carregando-se TH0 com 9Bh e TL0 com 14h (20 decimal).**
**A rotina de interrupção, além do tratamento desta, deverá escrever o valor inicial em TH0 (máx. contagem = 255 x 32)**

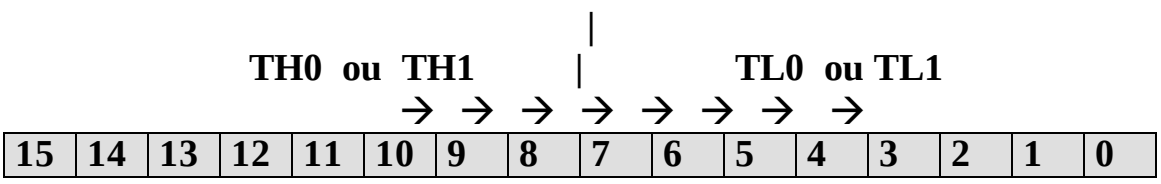
## MODO 1

- **Contador ou Temporizador de 16 bits.**
- **Selecionado fazendo-se: M1.x = 0 e M0.x = 1**
- **Temos um par de registradores TH0 e TL0 ou TH1 e TL1 escolhido para efetuar a contagem.**
- **É possível contagens de até FFFFh (65535 em decimal) com o valor inicial programável por software.**
- **Ao correr um overflow, o sistema recebe um pedido de interrupção interna que poderá ou não ser aceito.**



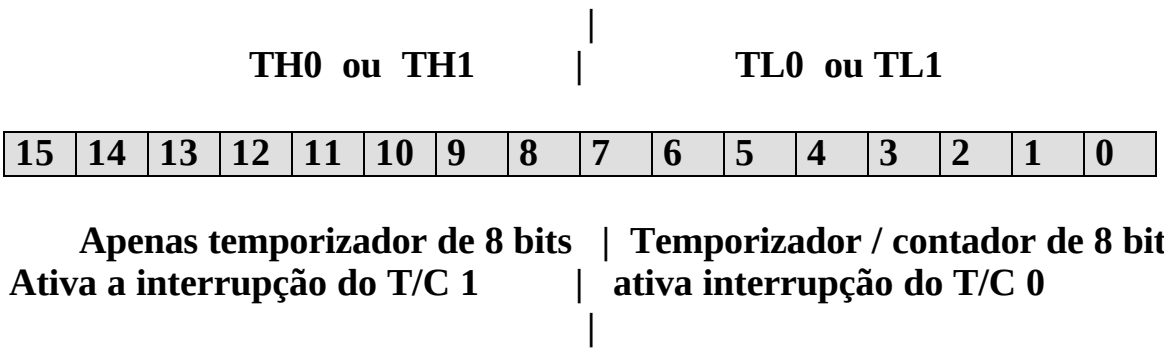

## MODO 2

- **Contador ou Temporizador de 8 bits com recarga automática.**
- **Selecionado fazendo-se: M1.x = 1 e M0.x = 0**
- **Nos registradores TL0 e TL1 ocorre a contagem.**
- **Nos registradores TH0 e TH1 temos os valores que serão carregados automaticamente nos registradores TL0 e TL1, sempre que ocorrer um overflow (interrupção), prosseguindo o sistema sob o comando do sinal externo (contador) ou interno (temporizador).**
- **Com este modo, temos um sistema no qual não é necessário escrever novamente via software o valor a ser contado.**
- **Todos os registradores podem ser alterados a qualquer momento pelo software, resultando em uma grande flexibilidade no trabalho com temporizações e contagens.**
- **O pedido de interrupção interna poderá ou não ser aceito pela CPU.**



## MODO 3

- **Contador de eventos de 8 bits e temporizador de 8 bits.**
- **Selecionado fazendo-se: M1.x = 1 e M0.x = 1**
- **Neste modo T/C 1 para sua operação e fica inerte, sem receber pulsos de contagem.**
- **Para T/C 0 temos dois sistemas de 8 bits, um em TH0 e outro em TL0.**
- **O T/C agora formado por TL0 será controlado pelos bits TR0 e TF0 do registrador TCON.**
- **O T/C agora formado em TH0 será controlado pelos bits TR1 e TF1 do registrador TCON.**
- **O T/C 0 será habilitado pelo bit de controle do T/C1 (TR1) e ao ocorrer um overflow de TH0, é o bit TF1, que será setado e não o TF0.**
- **Neste modo, o overflow em TH0 acionará o flag de requisição de interrupção referente ao T/C 1 e o overflow de TL0 acionará o flag de requisição de interrupção de T/C 0.**
- **Os pedidos de interrupções internas poderão ou não serem aceitos pela CPU.**



## 1.11 A COMUNICAÇÃO SERIAL:

### MODO SÍNCRONO DE COMUNICAÇÃO

- **Necessita de um sincronismo entre os sistemas de comunicação.**
- **Sincronismo é obtido através de um conjunto de bits, denominado "bits de sincronismo", que ao serem recebidos pelo elemento receptor, ajustam o seu relógio interno para possibilitar o recebimento de outro conjunto de bits referentes aos "dados".**
- **O transmissor envia um outro conjunto de bits chamado "bits de parada".**
- **Esses bits de parada podem conter ou não, informações sobre a confirmação do recebimento dos dados.**

| Caractere de sincronismo | | | | | | | bits de dados | | | | | | | bits de dados | | | | | | | caractere de final de transmissão | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 6 | 5 | 4 | 3 | 2 | 1 | 0 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 | 0 | x | x | x | x | x | x | x | x | x | x | X | x | x | x | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

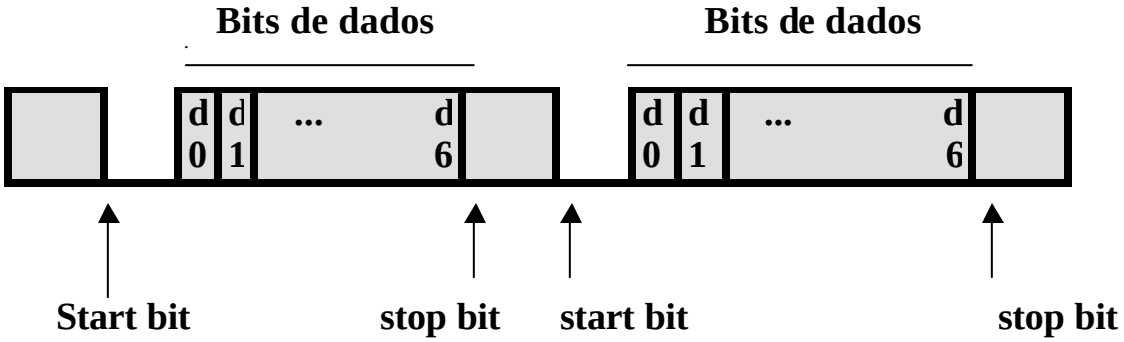
**OBS:**

**16h = SYN = SYNcronism idle ( Código ASCII )**

**04h = EOT = End Of Transmission ( Código ASCII )**

## MODO ASSÍNCRONO DE COMUNICAÇÃO

- **Não existe a necessidade de caracteres de sincronismo**
- **Para cada conjunto de bits de dados transmitidos temos também transmitidos um bit de início de transmissão (start bit) e um bit de final de transmissão (stop bit).**
- **O start bit é reconhecido pela transição de "1" para "0", na linha.**
- **Após o start bit, o sistema efetua periodicamente uma varredura na linha, associando os níveis lógicos encontrados aos bits dos dados de entrada.**
- **Ao reconhecer o sétimo bit, o sistema fica esperando o stop bit.**
- **O stop bit é a transição de "0" para "1", ou a permanência em "1" se a linha já se encontrava assim.**
- **Deve-se garantir que o transmissor e o receptor operem com a mesma taxa de comunicação.**




**OBS:**

**Os sinais de temporização e controle, utilizados em cada modo são gerados por um hardware específico para comunicação serial, sendo transparentes para o usuário.**

## CANAIS SIMPLEX, HALF-DUPLEX E FULL-DUPLEX

### Canal SIMPLEX

- **Este modo é constituído pela interligação de dispositivos no qual temos um elemento que apenas transmite e outro que apenas recebe.**

### Canal HALF-DUPLEX, ou SEMIDUPLEX

- **Neste modo de comunicação, temos elementos que recebem e transmitem dados, embora as duas operações não possam ocorrer simultaneamente.**

### Canal FULL-DUPLEX ou simplesmente DUPLEX

- **Consiste em um modo pelo qual os sistemas podem transmitir e receber dados simultaneamente.**

# A COMUNICAÇÃO SERIAL NO 8051

- **No 8051, a interface serial é do tipo Full-Duplex.**
- **O controle da transmissão e recepção de dados é feito por um registro denominado SBUF (serial Buffer – Endereço 99h).**
- **Uma escrita no SBUF implica em automática transmissão deste dado.**
- **Um dado ao "chegar" no pino correspondente será automaticamente recebido pelo sistema independentemente do usuário (p/ o canal serial habilitado e ajustado).**
- **Existem dois registradores SBUF, um destinado para a recepção e outro para a transmissão de dados.**
- **O reconhecimento é feito pelo sistema através das instruções que acessarão o mesmo, assim para uma instrução de escrita o registro de transmissão será alterado e para uma instrução de leitura o registro de recepção será acessado.**
- **A transmissão inicia-se assim que o dado é escrito no SBUF.**
- **A recepção é controlada pelo registro SCON.**

→ **SCON (Serial Control) – Endereço 98h**

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| SM0 | SM1 | SM2 | REN | TB8 | RB8 | T1 | R1 |

**SM0 e SM1 (Serial Mode):**

- **Estes dois bits controlam o funcionamento do canal serial.**

| SM0 | SM1 | Modo de Funcionamento | Taxa de transmissão |
|---|---|---|---|
| 0 | 0 | 0 | Fclock/12 |
| 0 | 1 | 1 | Variável |
| 0 | 0 | 2 | Fclock/32 ou Fclock/64 |
| 1 | 1 | 3 | Variável |

**SM2**

- **No modo 0, não impõe qualquer efeito no funcionamento do canal serial, devendo permanecer em 0.**
- **No modo 1, não produzirá pedido de interrupção se estiver setado e se o stop bit recebido for ilegal.**
- **Nos modos 2 e 3, habilita a comunicação entre várias cpu's 8051 p/=1.**

**REN (Reception Enable):**

- **Se estiver setado (=1), habilita a recepção tão logo um start bit seja detectado.**
- **Se estiver resetado (=0), desabilita a recepção, e o pino RXD pode ser usado como I/O.**

**TB8**

- **Nos modos 2 e 3, indica o estado do nono bit a ser transmitido.**
- **Pode ser setado ou resetado por software.**

**RB8**

- **Não é usado no modo 0.**
- **No modo 1 indica o estado do stop bit recebido, desde que SM2 seja igual a 0.**
- **Nos modos 2 e 3, indica o estado do nono bit de dados que foi recebido.**

**TI**

- **Flag de requisição de interrupção de transmissão.**
- **É setado pelo hardware após a transmissão do oitavo bit de dados quando no modo 0 e nos outros no início do stop bit.**
- **Permite que o programa de transmissão seja independente do programa principal, pois a cada final de transmissão a interrupção do canal serial pode gerenciar todo o processo, evitando assim os processos de varredura de alguns sistemas.**
- **Deve ser resetado pelo software da rotina de interrupção para permitir novas interrupções.**

**RI**

- **Flag de requisição de interrupção da recepção.**
- **É setado pelo hardware após a recepção do oitavo bit de dados quando no modo 0 e nos outros modos, ao moio do tempo de recpção do stop bit.**
- **Deve ser resetado pelo software da rotina de interrupção para permitir novas interrupções.**

# CAPÍTULO 2: SOFTWARE

## 2.1 MODOS DE ENDEREÇAMENTO

**A família 8051 possui os seguintes modos de endereçamento:**

**OBS:**

**Para diferenciar os modos de endereçamento e o tratamento dado a variáveis, constantes e endereços é utilizada a seguinte notação:**

**@ indica "Indireto"**
**# indica valor "Constante"**
**H indica que o valor é expresso em "Hexadecimal"**
**B indica que o valor é expresso em "Binário"**

**MODO IMEDIATO**

**Neste modo , temos um endereço de 8 bits logo após a instrução, no qual será efetuada a operação, dessa forma com ele podemos acessar apenas as primeiras 256 posições de memória (RAM interna e SFR).**

| | | |
|---|---|---|
| **Exemplos:** | **MOV A, 85H** | **A← (85H)** |
| | **MOV 86H, A** | **(86H)← A** |

**MODO REGISTRADOR**

**Neste modo, o nome do registrador a ser acessado está incluído no mnemônico (economia de um byte na memória de programa).**

**O registrador afetado (R0, R1, R2, R3, R4, R5, R6 ou R7) será o do banco de registradores selecionado no momento (RB0, RB1, RB2 ou RB3).**

**O endereçamento do banco de registradores é feito pelos bits RS2 e RS1, no registrador PSW, no instante de execução da instrução.**

| | | |
|---|---|---|
| **Exemplos:** | **MOV (98H), R5** | **(98H) ← R5** |
| | **MOV R2, (88H)** | **R2 ← (88H)** |

**MODO INDIRETO**

**Neste modo, o endereço sobre o qual a instrução atuará está indicado de forma indireta pelos registradores R0 ou R1, de qualquer um dos bancos de registradores, para endereços de 8 bits, ou indicado pelo registrador DPTR (Data Pointer, formado por DPH endereço 83H e DPL endereço 82H), para endereços de 16 bits.**

**Por esse modo de endereçamento é possível acessar a memória RAM interna ou externa.**

**Exemplos:** **MOV @R1, 43H** **(R1) ← (43H)**
**MOVX A, @DPTR** **A ← (DPTR/RAM ext.)**

**MODO ESPECÍFICO A REGISTRO**

**Neste modo, o registrador em questão, já faz parte do mnemônico da instrução.**

**Exemplos:** **DA A** **ajuste decimal de A**
**CRL A** **A ← 00H**

**MODO CONSTANTE IMEDIATA**

**Este modo permite trabalhar com uma constante de forma direta.**

**Exemplos:** **MOV A, #44H** **A ← 44H**
**MOV R7, #63H** **R7 ← 63H**

**MODO INDEXADO**

**Neste modo, o endereço efetivo é a soma do Acumulador e de um registrador de 16 bits, que poderá ser o PC ou o DPTR.**

**Esse modo de endereçamento tem como principal aplicação a leitura de tabelas presentes na memória ROM, transferindo-as para a memória RAM e/ou realizando processamento sobre esses dados.**

**Exemplos:** **MOVC A, @A+DPTR** **A ← (A+DPTR)**
**JMP @A+DPTR** **PC ← A+DPTR**

**MODO DESVIO INDEXADO**

**Esse modo é usado por instruções de desvio condicionais, que somam ao valor do PC já ajustado, o dado de 8 bits presente no final da instrução.**

**Exemplo:** **JZ 15H** **Desvio relativo se A=00**

# EXEMPLOS DA ESCRITA DE ALGUMAS INSTRUÇÕES

| Mnemônico | Descrição |
|---|---|
| ADD A, @R1 | A ← A + (R1) |
| ADDC A, #58 | A ← A + 58 + carry |
| SUBB A, R0 | A ← A – R0 – carry |
| INC DPTR | DPTR ← DPTR + 1 |
| MUL AB | B A ← A x B |
| DIV AB | A ← A/B  B ← resto |
| ANL 58, #66 | (58) ← (58) [and] #66 |
| XRL A, @R1 | A ← A [exclusive or] (R1) |
| RL A | d7←d6←d5←d4←d3←d2←d1←d0←d7 |
| RRC A | cy→d7→d6→d5→d4→d3→d2→d1→d0→cy |
| SWAP A | em A: d7 d6 d5 d4 ←→ d3 d2 d1 d0 |
| MOV A, 1Fh | A ← (1Fh) |
| MOV A, #3Ch | A ← #3Ch ;hexadecimal |
| MOV A, #10101110b | A ← #10101110b ;binário |
| MOV A, #78 | A ← 78 ;decimal |
| MOV @R0, #0F2h | (R0) ← F2h { atenção para o caractere “0” } |
| MOV DPTR, #2000h | DPTR ← 2000h |
| MOVC A, @A+DPTR | A ← (A + DPTR) ;única forma de ler uma constante na memória de programa |
| MOVX A, @R0 | A ← (R0) ;leitura da RAM externa |
| MOVX @DPTR, A | (DPTR) ← A ;escrita na RAM externa |
| XCH A, 28 | A ←→ (28) |
| XCHD A, @R0 | Para: A = 21h e (R0) = 4Dh, após a instrução temos: A = 2Dh e (R0) = 41h |
| POP 70 | (70) ← (SP)<br>SP ← SP + 1 |
| PUSH 70 | SP ← SP – 1<br>(SP) ← (70) |
| CLR EX1 | bit EX1 ←0 |
| CLR ABh | Bit ABh ← 0 |
| CPL A | A ← complemento de A |
| ORL C, 1Ah | cy (or) (bit 1Ah)’ |
| JNB P1.1, 2Ch | se P1.1 = 0 ;PC ← PC + 2Ch o programa desvia<br>se P1.1 =1 ;programa prossegue |

## 2.2 TABELAS DE INSTRUÇÕES DA FAMÍLIA 8051

**Para diferenciar os modos de endereçamento, o tratamento dado a variáveis, constantes e endereços, registradores e seus conteúdos, além de valores numéricos genéricos de diferentes formatos e para diversas aplicações, é utilizada por alguns fabricantes a seguinte notação:**

| | |
|---|---|
| **Rn** | **Indica registrador R0 a R7 genericamente, dependendo de "n".** |
| **Ri** | **Indica o registrador R0 ou R1 genericamente, dependendo de "i".** |
| **@** | **Significa "endereçado pelo valor de ... "** |
| **#dado8** | **Indica valor constante de 8 bits.** |
| **#dado16** | **Indica valor constante d 16 bits.** |
| **direto** | **Indica um endereço de memória de 8 bits (primeiras 256 posições internas ou externas).** |
| **rel** | **Indica que o endereço é relativo.** |
| **end2k** | **Indica endereço dentro de uma faixa de 2Kbytes.** |
| **end16** | **Indica um endereço de 16 bits.** |
| **bit** | **Indica um bit individualmente endereçado.** |

### CICLOS DE MÁQUINA

**Corresponde a um período de tempo equivalente a 12 períodos do clock, portanto para um microcontrolador com um cristal de 12M Hz, um ciclo de máquina tem a duração de $1 \times 10^{-6}$ s.**

**Todas as instruções sempre são executadas em um número inteiro de ciclos de máquina, que na família 8051 são de 1, 2 ou 4 ciclos.**

## INSTRUÇÕES PARA TRANSFERÊNCIA DE DADOS

| Mnemônico | Descrição | Nº de bytes | Nº de pulsos de clock |
|---|---|---|---|
| MOV A, Rn | Move o registrador para o acumulador | 1 | 12 |
| MOV A, direto | Move a memória para o acumulador | 2 | 12 |
| MOV A, @Ri | Move RAM endereçada por Ri para o acumulador | 1 | 12 |
| MOV A, #dado8 | Move o dado para o acumulador | 2 | 12 |
| MOV Rn, A | Move o acumulador para o registrador | 1 | 12 |
| MOV Rn, direto | Move a memória para o registrador | 2 | 12 |
| MOV Rn, #dado8 | Move o dado para o registrador | 2 | 12 |
| MOV direto, A | Move o acumulador para a memória | 2 | 12 |
| MOV direto, Rn | Move o registrador para a memória | 2 | 24 |
| MOV direto1, direto2 | Move o conteúdo da memória direta2 para a memória direta1 | 3 | 24 |
| MOV direto, @Ri | Move RAM endereçada por Ri para a memória | 2 | 24 |
| MOV direto, #dado8 | Move o dado para a memória | 3 | 24 |
| MOV @Ri, A | Move o acumulador para a RAM endereçada por Ri | 1 | 12 |
| MOV @Ri, direto | Move a memória para a RAM endereçada por Ri | 2 | 24 |
| MOV @Ri, #dado8 | Move o dado para a RAM indireta | 2 | 12 |
| MOV DPTR, #dado16 | Move dado de 16 bits para o DPTR | 3 | 24 |
| MOVC A, @A + DPTR | Soma: A + DPTR obtendo um endereço de 16 bits na memória de programa e carrega acumulador com esta memória | 1 | 24 |
| MOVC A, @A + PC | Idem a anterior mas soma A + PC | 1 | 24 |
| MOVX A, @Ri | Move RAM externa (endereço de 8 bits) para o acumulador | 1 | 24 |
| MOVX A, @DPTR | Move RAM externa (endereço de 16 bits) para o acumulador | 1 | 24 |
| MOVX @Ri, A | Move acumulador para a RAM externa (endereço de 16 bits) | 1 | 24 |
| MOVX @DPTR, A | Move acumulador para a RAM externa (endereço de 16 bits) | 1 | 24 |
| PUSH direto | Incrementa o SP e então coloca a memória no stack | 2 | 24 |
| POP direto | Retira o dado do stack e o coloca na memória, depois decrementa o SP | 2 | 24 |
| XCH A, Rn | Troca os conteúdos do acumulador e do registrador | 1 | 12 |
| XCH A, direto | Troca a memória com o conteúdo do acumulador | 2 | 12 |
| XCH A, @Ri | Troca RAM indireta com o conteúdo do acumulador | 1 | 12 |
| XCHD A, @Ri | Troca o nibble menos significativo do acumulador e da RAM indireta entre si | 1 | 12 |

## INSTRUÇÕES ARITMÉTICAS

| Mnemônico | Descrição | Nº de bytes | Nº de pulsos de clock |
|---|---|---|---|
| **ADD A, Rn** | **Soma o conteúdo de Rn ao acumulador. Resultado em A.** | **1** | **12** |
| **ADD A, direto** | **Soma o conteúdo da posição de memória ao acumulador. Resultado em A.** | **2** | **12** |
| **ADD A, @Ri** | **Soma o conteúdo da RAM endereçada por Ri ao acumulador. Resultado em A.** | **1** | **12** |
| **ADD A, #dado8** | **Soma o dado ao acumulador. Resultado em A.** | **2** | **12** |
| **ADDC A, Rn** | **Soma o conteúdo de Rn e o carry ao acumulador. Resultado em A.** | **1** | **12** |
| **ADDC A, direto** | **Soma o conteúdo da posição de memória e o carry ao acumulador. Resultado em A.** | **2** | **12** |
| **ADDC A, @Ri** | **Soma o conteúdo da RAM endereçada por Ri e o carry ao acumulador. Resultado em A.** | **1** | **12** |
| **ADDC A, #dado8** | **Soma o dado e o carry ao acumulador. Resultado em A.** | **2** | **12** |
| **SUBB A, Rn** | **Subtrai o conteúdo de Rn e o borrow do acumulador. Resultado em A.** | **1** | **12** |
| **SUBB A, direto** | **Subtrai o conteúdo da posição de memória e o borrow do acumulador. Resultado em A.** | **2** | **12** |
| **SUBB A, @Ri** | **Subtrai o conteúdo da RAM endereçada por Ri e o borrow do acumulador. Resultado em A.** | **1** | **12** |
| **SUBB A, #dado8** | **Subtrai o dado e o borrow do acumulador. Resultado em A.** | **2** | **12** |
| **INC A** | **Soma 1 ao acumulador.** | **1** | **12** |
| **INC Rn** | **Soma 1 ao conteúdo de Rn** | **1** | **12** |
| **INC direto** | **Soma 1 a posição de memória** | **2** | **12** |
| **INC @Ri** | **Soma 1 a RAM endereçada por Ri** | **1** | **12** |
| **DEC A** | **Subtrai 1 do acumulador.** | **1** | **12** |
| **DEC Rn** | **Subtrai 1 do conteúdo de Rn** | **1** | **12** |
| **DEC direto** | **Subtrai 1 da posição de memória** | **2** | **12** |
| **DEC @Ri** | **Subtrai 1 da RAM endereçada por Ri** | **1** | **12** |
| **INC DPTR** | **Soma 1 ao registro DPTR** | **1** | **24** |
| **MUL AB** | **Multiplica A e B, resultado em BA** | **1** | **48** |
| **DIV AB** | **Divide A por B, resultado em A, reto em B** | **1** | **48** |
| **DA A** | **Ajuste decimal do acumulador.** | **1** | **12** |

## INSTRUÇÕES LÓGICAS

| Mnemônico | Descrição | Nº de bytes | Nº de pulsos de clock |
|---|---|---|---|
| ANL A, Rn | "E" entre o registrador e o acumulador. Resultado em A. | 1 | 12 |
| ANL A, direto | "E " entre a memória e o acumulador. Resultado em A. | 2 | 12 |
| ANL A, @Ri | "E" entre RAM indireta e o acumulador. Resultado em A. | 1 | 12 |
| ANL A, #dado8 | "E" entre o dado e o acumulador. Resultado em A. | 2 | 12 |
| ANL direto, A | "E" entre acumulador e memória. Resultado na memória. | 2 | 12 |
| ANL direto, #dado8 | "E " entre dado e memória. Resultado na memória | 3 | 24 |
| ORL A, Rn | "OU" entre o registrador e o acumulador. Resultado em A. | 1 | 12 |
| ORL A, direto | "OU " entre a memória e o acumulador. Resultado em A. | 2 | 12 |
| ORL A, @Ri | "OU" entre RAM indireta e o acumulador. Resultado em A. | 1 | 12 |
| ORL A, #dado8 | "OU" entre o dado e o acumulador. Resultado em A. | 2 | 12 |
| ORL direto, A | "OU" entre acumulador e memória. Resultado na memória | 2 | 12 |
| ORL direto, #dado8 | "OU " entre dado e memória. Resultado na memória | 3 | 24 |
| XRL A, Rn | "XOR" entre o registrador e o acumulador. Resultado em A. | 1 | 12 |
| XRL A, direto | "XOR " entre a memória e o acumulador. Resultado em A. | 2 | 12 |
| XRL A, @Ri | "XOR" entre RAM indireta e o acumulador. Resultado em A | 1 | 12 |
| XRL A, #dado8 | "XOR"entre o dado e o acumulador.Resultado em A | 2 | 12 |
| XRL direto, A | "XOR" entre acumulador e memória. Resultado na memória | 2 | 12 |
| XRL direto, #dado8 | "XOR " entre dado e memória.Resultado na memória | 3 | 24 |
| CRL A | Faz A = 0 | 1 | 12 |
| CPL A | Inverte o estado de todos os bits do acumulador | 1 | 12 |
| RL A | Desloca o acumulador um bit a esquerda<br>d7 ← d6 ← d5 ← d4 ← d3 ← d2 ← d1 ← d0 ← d7 | 1 | 12 |
| RLC A | Desloca o acumulador um bit a esquerda pela carry<br>CY ← d7 ← d6 ← d5 ← d4 ← d3 ← d2 ← d1 ← d0 ← CY | 1 | 12 |
| RR A | Desloca o acumulador um bit a direita<br>d7 → d6 → d5 → d4 → d3 → d2 → d1 → d0 → d7 | 1 | 12 |
| RRC A | Desloca o acumulador um bit a direita pela carry<br>d7 → d6 → d5 → d4 → d3 → d2 → d1 → d0 → CY → d7 | 1 | 12 |
| SWAP A | Troca os nibbles + e – significativos do acumulador | 1 | 12 |

## INSTRUÇÕES DE DESVIO

| Mnemônico | Descrição | Nº de bytes | Nº de pulsos de clock |
|---|---|---|---|
| **ACALL end2K** | **Chama sub-rotina numa contida faixa de 2Kbytes da posição atual (dentro do espaço de –1024 a +1024 posições de memória).** | **2** | **24** |
| **LCALL end16** | **Chama sub-rotina em qualquer posição da memória de programa (até 64K bytes).** | **3** | **24** |
| **RET** | **Retorna da sub-rotina.** | **1** | **24** |
| **RETI** | **Retorna da interrupção.** | **1** | **24** |
| **AJMP end2K** | **Desvia para endereço dentro de uma faixa de 2Kbytes da posição atual (dentro do espaço de –1024 a +1024 posições de memória).** | **2** | **24** |
| **LJMP end16** | **Desvia para qualquer posição da memória de programa (até 64K bytes).** | **3** | **24** |
| **SJMP rel** | **Desvio curto relativo em uma faixa de 255 bytes (dentro do espaço –128 a +127 posições de memória).** | **2** | **24** |
| **JMP @A + DPTR** | **Desvia para o endereço obtido da soma do acumulador com o DPTR.** | **1** | **24** |
| **JZ rel** | **Desvia se o acumulador "for zero".** | **2** | **24** |
| **JNZ rel** | **Desvia se o acumulador "não for zero".** | **2** | **24** |
| **CJNE A, direto, rel** | **Compara e desvia se o acumulador for diferente da memória endereçada.** | **3** | **24** |
| **CJNE A, #dado8, rel** | **Compara e desvia se o acumulador for diferente do dado.** | **3** | **24** |
| **CJNE Rn, #dado8, rel** | **Compara e desvia se o registrador for diferente do dado..** | **3** | **24** |
| **CJNE @Ri, #dado8, rel** | **Compara e desvia se a RAM indireta for diferente do dado.** | **3** | **24** |
| **DJNZ Rn, rel** | **Decrementa o registrador e desvia se for "diferente de zero".** | **2** | **24** |
| **DJNZ direto, rel** | **Decrementa a memória e desvia se for "diferente de zero".** | **3** | **24** |
| **NOP** | **Nenhuma operação.** | **1** | **12** |

## INSTRUÇÕES PARA VARIÁVEIS BOOLEANAS

| Mnemônico | Descrição | Nº de bytes | Nº de pulsos de clock |
|---|---|---|---|
| **CLR C** | **Zera o carry flag** | **1** | **12** |
| **CLR bit** | **Zera o bit endereçado** | **2** | **12** |
| **SETB C** | **Seta o carry flag** | **1** | **12** |
| **SETB bit** | **Seta o bit endereçado** | **2** | **12** |
| **CPL C** | **Inverte o estado do carry flag** | **1** | **12** |
| **CPL bit** | **Inverte o estado do bit endereçado** | **2** | **12** |
| **ANL C, bit** | **"E" entre o carry flag e o bit endereçado. Resultado no carry flag.** | **2** | **24** |
| **ANL C, /bit** | **"E" entre o carry flag e o complemento do bit endereçado. Resultado no carry flag.** | **2** | **24** |
| **ORL C, bit** | **"OU" entre o carry flag e o bit endereçado. Resultado no carry flag.** | **2** | **24** |
| **ORL C, /bit** | **"OU" entre o carry flag e o complemento do bit endereçado. Resultado no carry flag.** | **2** | **24** |
| **MOV C, bit** | **Move o bit endereçado para o carry flag.** | **2** | **12** |
| **MOV bit, C** | **Move o carry flag para o bit endereçado.** | **2** | **24** |
| **JC rel** | **Desvia se o carry flag estiver setado.** | **2** | **24** |
| **JNC rel** | **Desvia se o carry flag estiver zerado.** | **2** | **24** |
| **JB bit, rel** | **Desvia se o bit endereçado estiver setado.** | **3** | **24** |
| **JNB bit, rel** | **Desvia se o bit endereçado estiver zerado.** | **3** | **24** |
| **JBC bit, rel** | **Desvia se o bit endereçado estiver setado e depois zera o bit.** | **3** | **24** |

# CAPÍTULO 3: PROGRAMAÇÃO E SIMULAÇÃO

## 3.1 “COMPILAÇÃO” E “LINKAGEM”

**Um programa escrito em linguagem assembly (PROGRAMA FONTE), não pode ser diretamente processado pelo microcontrolador do sistema, devendo primeiramente ser traduzido para a sua linguagem de máquina, com o uso de tabelas ou através de um programa destinado para tal tarefa chamado de COMPILADOR, que fornece então como saída o PROGRAMA OBJETO.**

**Define-se COMPILADOR como um programa aplicativo que transforma um arquivo constituído por códigos ASCII (PROGRAMA FONTE: obrigatoriamente com extensão “.ASM”), gerado normalmente por um editor de textos, em um arquivo binário que contém os bytes correspondentes às instruções (códigos de máquina) do microcontrolador.**

**Como resultado da compilação são criados dois arquivos:**

- **Arquivo de mesmo nome porém com a extensão “.OBJ” (PROGRAMA OBJETO).**
- **Arquivo de mesmo nome, porém com a extensão “.PRN”, que corresponde a um arquivo texto que mostra o resultado da compilação, contendo para cada linha de programa, o código de máquina correspondente à instrução, sendo muito útil na depuração de erros de compilação.**

**A “linkagem” tem a finalidade de reunir, em um único arquivo, todos as rotinas escrita em um ou vários arquivos diferentes.**

**Após a “linkagem” são gerados dois arquivos:**

- **Arquivo de mesmo nome, sem extensão, usado pelo programa simulador.**
- **Arquivo de mesmo nome, porém com a extensão “.HEX”, usado por gravadores de memórias e microcontroladores e também pelo programa simulador.**

**PROGRAMA → COMPILADOR → PROGRAMA → LINKER → PROGRAMA**
**FONTE OBJETO EXECUTÁVEL**
**.ASM .OBJ .HEX**

**O compilador, o “Linker” e o Simulador usados no laboratório são produzidos pela AVOCET.**

## 3.2 DIRETIVAS (ou PSEUDO-INSTRUÇÕES)

**Além das instruções pertencentes ao microcontrolador em questão, a linguagem assembly possui ainda algumas instruções especiais, pseudo-instruções ou diretivas, que são usadas apenas para a estruturação do programa.**

**Estas instruções especiais, que não são traduzidas para o código de máquina por não pertencerem ao conjunto de instruções do microcontrolador escolhido, possuem apenas funções especiais no programa como: definir símbolos, estabelecer o endereço inicial do programa, reservar área de memória etc, não sendo portanto, processadas.**

### PSEUDO-INSTRUÇÕES MAIS USADAS

**ORG (ORIGIN)**

**Formato:**

| Pseudo-instrução | operando |
|---|---|
| ORG | endereço |

**Função:**

**Usado para o endereço inicial de memória, no qual o programa ou um trecho de programa será armazenado.**

**Exemplo:**

| | |
|---|---|
| ORG | 0100H |

**Assim o programa objeto será carregado na memória a partir do endereço 0100H**

**DB (DEFINE BYTE)**

**Formato:**

| Pseudo-instrução | operando |
|---|---|
| DB | byte |

**Função:**

**O byte do operando é carregado diretamente na posição de memória escolhida pelo ORG.**

**Exemplo:**

| | |
|---|---|
| ORG | 0050H |
| DB | 3FH |
| DB | 1AH |

**Assim os bytes 3FH e 1AH foram armazenados nas posições de memória 0050H e 0051H respectivamente.**

**END**

**Formato:**

| Pseudo-instrução |
|---|
| END |

**Função:**

**Indica o final do programa.**

## 3.3 USO DO COMPILADOR E DO LINKER (PASSO A PASSO)

**OBS:**

**Comandos digitados na linha de ”prompt do DOS”.**

1. **Com o “NE” ou “EDIT” escrever um programa em assembly nomeado obrigatoriamente com a extensão “.ASM” (criando assim programa fonte).**

   **OBS: No editor de textos deve-se obrigatoriamente reservar as colunas de “1” a “6” para os “labels” ou “tags” que representam os endereços do programa, de entrada de “loops”, chamada de sub-rotinas etc.**

2. **Com o “X8051” (compilador) obter os arquivos com extensão “.OBJ” (programa objeto) e “.LST” (listagem) da seguinte forma:**

| | |
|---|---|
| **X8051** | **[ENTER]** |
| **Listing Destination (.....): D** | **[ENTER]** |
| **Generate cross reference** | **[ENTER]** |
| **Input file name:.....nome do arquivo.ASM** | **[ENTER]** |
| **Output file name: ..... nome do arquivo** | **[ENTER]** |

3. **Com o “LINK” ligar o arquivo “.OBJ”, gerando um arquivo “.HEX” da seguinte forma:**

| | |
|---|---|
| **LINK** | **[ENTER]** |
| **Input file name: ..... .OBJ** | **[ENTER]** |
| **Enter offset for .....** | **[ENTER]** |
| **Input file name:** | **[ENTER]** |
| **Output file name:** | **[ENTER]** |
| **Options (.....): H** | **[ENTER]** |

4. **No simulador:**
   **Load**
   **Avocet**
   **.... .HEX**
   **Simulation**
   - **F1 (total)**
   - **F10 (passo a passo)**

## 3.4 SIMULADOR DOS MICROCONTROLADORES DA FAMÍLIA 8051

******** TELA DE ENTRADA DO SIMULADOR: ********

---

**AVSIM 8051 Simulator/Debugger**

**Licensed by Avocet Systems, Inc.**

**To receive updates, see 'Help Registration'**



**Intel 8051 Family Microcomputers**

| HMOS ROM | HMOS ROMless | CMOS ROM |
|---|---|---|
| A: 8051/8751 | C: 8031 | E: 80C51 |
| B: 8052/8752 | D: 8032 | F: 80C31 |

**Choose a CPU for simulation:**

---

******** DEVE-SE SELECIONAR UMA OPÇÃO EM FUNÇÃO ********
********DA CPU ESCOLHIDA PARA O PROJETO ********

## ****** TELA DE SIMULAÇÃO: ******

```
LABEL    OPERATION        8051/8751 AVSIM 8051 Simulator/Debugger       V1.6
0000H   no    memory      CPUREGISTERS FLAGS      SCL SPD DSP SKPCURSOR
0001H   no    memory      C  Accumulator  AC F0 OV P OFF HI   ON  OFF  MENU
0002H   no    memory      0  00000000:00:_ 0    0    0    0                Cycles:
0003H   no    memory      EA:
0004H   no    memory      PC:0000 »FF  FF FF FF TimersTH/TL TF /TR G/T/M1/M0
0005H   no    memory      SP:   07 » 00  00  00  00  T0:      00   00    0    0   0   0    0    0
0006H   no    memory                 00  00  00  00  T1:      00   00    0    0   0   0    0    0
0007H   no    memory      DP:0000 »FF FF  FF FF
0008H   no    memory      R0:00:_ »  00:_   RB:00  Ints  A  S T1  X1   T0 X0 Edg IT  IE
0009H   no    memory      R1:00:_ »  00:_    B:00   En    0  0 0    0    0    0   X0:    0   0
000AH   no    memory      R2:00   R4:00     R6:00  Pr    0  0 0    0    0        X1:    0   0
000BH   no    memory      R3:00   R5:00     R7:00  SBUF:  In    Out   PCON:0xxxxxxx
000CH   no    memory      Data S pace                    00:_  00:_   SCON:00000000
000DH   no    memory      00    00 00 00 00 00 00 00 00 _____________ Ports
000EH   no    memory      08    00 00 00 00 00 00 00 00 _____________ P0     11111111
000FH   no    memory      10    00 00 00 00 00 00 00 00 _____________ FF:_:11111111
0010H   no    memory      18    00 00 00 00 00 00 00 00 _____________ P1     11111111
0011H   no    memory      Data Space                                  FF:_:11111111
0012H   no    memory      20    00 00 00 00 00 00 00 00 _____________ P2     11111111
0013H   no    memory      28    00 00 00 00 00 00 00 00 _____________ FF:_:11111111
0014H   no    memory      30    00 00 00 00 00 00 00 00 _____________ P3     11111111
0015H   no    memory      38    00 00 00 00 00 00 00 00 _____________ FF:_:11111111
>Select Command - or use arrow keys
Dump  Expression  commandFile  Help  IO  Load        --space--  ESC to screen
```

## OPERAÇÃO BÁSICA DO SIMULADOR

**Para uma rápida ambientação com o programa simulador, é apresentada uma seqüência de testes para as principais teclas e/ou comandos:**

- **Visualização geral da tela, tentando reconhecer os seus principais campos (coloridos) e funções correspondentes.**
- **Teclas:** **ESC**
  **CTRL C**

- **Comandos:** **Help →** **Commands**
  **Display**
  **Simulation**
  **Avocet**
  **Registration**

  **Load →** **Avocet**

  **Quit →** **Exit**

  **setUp →** **Cursor →** **Yes**
  **No**

  **View →** **Memory-map**
  **Symbols →** **Alpha**
  **Registers**
  **Data**
  **Bit**
  **SFR**

  **eXecute**

- **Seqüência para carregar um arquivo no simulador e informações sobre a simulação:**
  **Load → Avocet → ... .OBJ** **Help → Simulation**

- **Teclas para a simulação**
  **F1 :executa o programa inteiro.**
  **F10 :executa um programa instrução por instrução.**
  **F9 :volta para a condição anterior, após a execução de uma instrução.**

## 3.5 EXERCÍCIOS / EXEMPLOS DE PROGRAMAÇÃO - APLICAÇÕES PARA O SIMULADOR AVSIM 8051

**OBJETIVOS:**

1. Praticar o uso do programa simulador da família de microcontroladores estudada.
2. Ampliar o conjunto de instruções e pseudo-instruções (diretivas) conhecido pelos alunos.
3. Explorar recursos do microcontrolador como: interrupções, contadores etc.
4. Obter respostas com estruturas próximas as empregadas em sistemas reais microcontrolados, equivalendo ao FIRMWARE destes (SOFTWARE armazenado exclusivamente em memória não volatil, e com a função de controlar o HARDWARE ).
5. Em todos os enunciados (dos exercícios seguintes) fica subentendido a seguinte frase: "Escrever, compilar e simular um programa em linguagem assembly do microcontrolador 8751 para...".

**EXEMPLO / EXERCÍCIO 1**

Escrever e simular um programa para o microcontrolador 8751 que execute as seguintes tarefas:

A. Inicialmente carregue os registradores R0, R1, R2, R3, R4, R5, R6 e R7 com os valores: 00H, 10H, 20H, 30H, 40H, 50H, 60H e 70H respectivamente.
B. Envie os bytes 00H para P0, FFH para P1, 0FH para P2 e AAH para P3.
C. Incremente o conteúdo dos registradores R0, R1, R2, R3, R4, R5, R6 e R7.
D. Envie os bytes FFH para P0, 00H para P1, F0H para P2 e 55H para P3.
E. Volte ao passo C (entrando em loop).

OBS:

- Atenção com o uso das diretivas "ORG" e "END" e dos label "L1".
- Para facilitar o entendimento e a depuração, procure sempre manter uma boa estética na digitação do programa.
- Testar a solução apresentada procurando ambientar-se com a operação do simulador:

```
;*************************************************************************
;       EXEMPLO / EXERCÍCIO 1 – ESCRITA EM REGISTRADORES E NAS PORTAS
;*************************************************************************
            ORG     0000H
            LJMP    INICIO          ;desvia dos endereços reservados p/ as interrupções

            ORG     0050H
INICIO:     MOV     R0, #00H        ;carrega o registrador R0
            MOV     R1, #10H        ;carrega o registrador R1
            MOV     R2, #20H        ;carrega o registrador R2
            MOV     R3, #30H        ;carrega o registrador R3
            MOV     R4, #40H        ;carrega o registrador R4
            MOV     R5, #50H        ;carrega o registrador R5
            MOV     R6, #60H        ;carrega o registrador R6
            MOV     R7, #70H        ;carrega o registrador R7

L1:         MOV     P0, #00H        ;envia byte para a porta P0
            MOV     P1, #0FFH       ;envia byte para a porta P1
            MOV     P2, #0FH        ;envia byte para a porta P2
            MOV     P3, #0AAH       ;envia byte para a porta P3

            INC     R0              ;incrementa o conteúdo do registrador R0
            INC     R1              ;incrementa o conteúdo do registrador R1
            INC     R2              ;incrementa o conteúdo do registrador R2
            INC     R3              ;incrementa o conteúdo do registrador R3
            INC     R4              ;incrementa o conteúdo do registrador R4
            INC     R5              ;incrementa o conteúdo do registrador R5
            INC     R6              ;incrementa o conteúdo do registrador R6
            INC     R7              ;incrementa o conteúdo do registrador R7

            MOV     P0, #0FFH       ;envia byte para a porta P0
            MOV     P1, #00H        ;envia byte para a porta P1
            MOV     P2, #0F0H       ;envia byte para a porta P2
            MOV     P3, #055H       ;envia byte para a porta P3
            LJMP    L1
            END
```

**EXEMPLO / EXERCÍCIO 2**

Escrever e simular um programa para o microcontrolador 8751 com o objetivo de controlar os bytes fornecidos para as portas P0 e P1 (saídas do sistema) através do status do bit P3.7 (entrada do sistema), da seguinte forma:

| Enquanto a entrada P3.7 for = 0 | Enquanto a entrada P3.7 for = 1 |
|---|---|
| Enviar para a porta P0 alternadamente os bytes 00h e FFh, separados de um pequeno "delay". Deixa a porta P1 inalterada. | Enviar para a porta P1 alternadamente os bytes 55h e AAh, separados de um pequeno "delay". Deixa a porta P0 inalterada. |

OBS:

- Atenção com o uso das diretivas "ORG" e "END" e dos labels "L1", "L2" e "DELAY".
- Verifique o uso da instrução de "teste de bit" e de chamada e retorno de sub-rotina.
- Para facilitar o entendimento e a depuração, procure sempre manter uma boa estética na digitação do programa.
- Testar a solução apresentada procurando ambientar-se com a operação do simulador:

```
;*************************************************************
;       EXEMPLO / EXERCÍCIO 2 – TESTE DE BITS
;*************************************************************
                ORG     0000H
                LJMP    INICIO          ;desvia dos endereços
                                        ;reservados p/ as interrupções
                ORG     0050H
INICIO:         JNB     P3.7, L1        ;desvia se bit de controle = 0

                MOV     P1, #55H        ;controla a porta P1
                ACALL   DELAY
                MOV     P1, #0AAH
                ACALL   DELAY
                LJMP    INICIO

L1:             MOV     P0, #00H        ;controla a porta P0
                ACALL   DELAY
                MOV     P0, #0FFH
                ACALL   DELAY
                LJMP    INICIO

DELAY:          MOV     A, #0FH         ;estabelece um pequeno
L2:             DEC     A               ;atraso (para facilitar a
                JNZ     L2              ;visualização no simulador.
                RET                     ;se necessário alterar o atraso.

                END
```

**EXERCÍCIO 3**

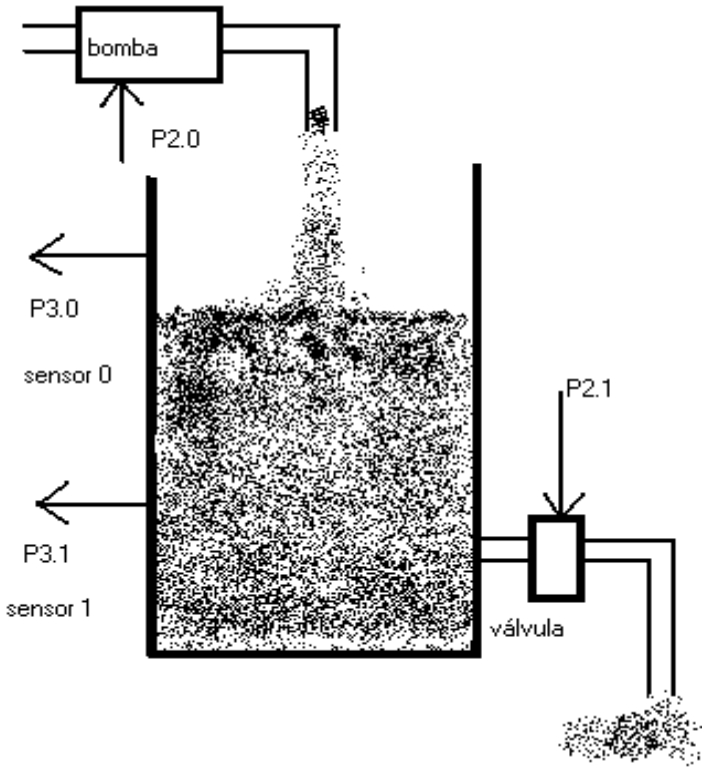
Escrever um programa para controlar o reservatório da figura a seguir, com o objetivo de manter o nível do líquido deste, que é consumido pelo sistema, entre a posição dos sensores 0 e 1.

Definição das variáveis de entrada e saída:

| | | | |
|---|---|---|---|
| sensor 0 = 1 | → líquido detectado | sensor 0 = 0 | → caso contrário |
| sensor 1 = 1 | → líquido detectado | sensor 1 = 0 | → caso contrário |
| bomba = 1 | → acionada | bomba = 0 | → desligada |
| válvula = 1 | → aberta | válvula = 0 | → fechada |

OBS:

- Para a resolução deste problema recomenda-se inicialmente a elaboração de um fluxograma ou algoritmo e após, a sua codificação em assembly.
- O enunciado é propositadamente aberto, assim qualquer dado ou condição adicional pode ser adotada pelo programador.




**EXERCÍCIO 4**

Escrever um programa para controlar a operação das duas esteiras da figura, com os seguintes passos:

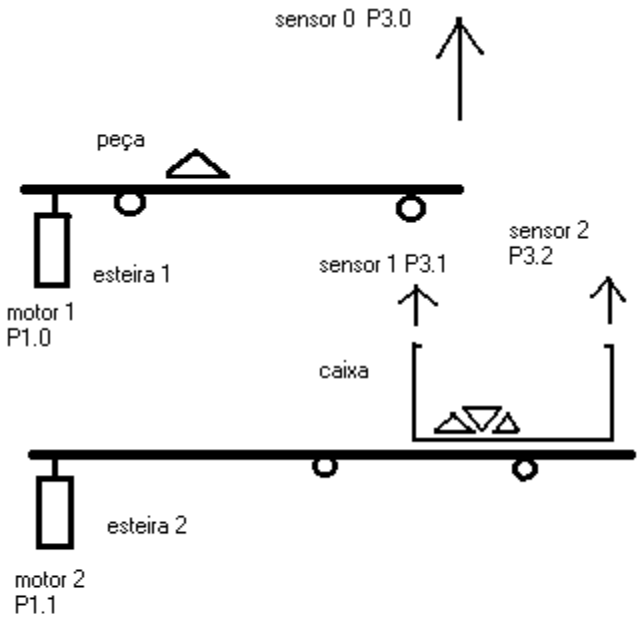
A. Posicionar a caixa na esteira 2.
B. Acionar a esteira 1 para encher a caixa com um total de 20 peças.
C. Retirar a caixa cheia e posicionar uma nova.

Definição das variáveis de entrada e saída:

| | | | |
|---|---|---|---|
| sensor 0 = 1 | → peça detectada | sensor 0 = 0 | → caso contrário |
| sensor 1 = 1 | → caixa detectada | sensor 1 = 0 | → caso contrário |
| sensor 2 = 1 | → caixa detectada | sensor 2 = 0 | → caso contrário |
| motor 1 = 1 | → motor ligado | motor 1 = 0 | → motor desligado |
| motor 2 = 1 | → motor ligado | motor 2 = 0 | → motor desligado |

OBS:

- O emprego dos sensores 1 e 2 evita que uma eventual posição intermediária da caixa seja aceita como correta (por exemplo, no momento em que o sistema é ligado).
- Pela diferença de dimensões entre o objeto e a área ativa do sensor, deve-se evitar que uma peça seja contada mais de uma vez, ao passar na frente do sensor 0.
- O enunciado é propositadamente aberto, assim qualquer dado ou condição adicional pode ser adotada pelo programador.




**EXERCÍCIO 5**

Escrever um programa que execute os seguintes passos:

A. Inicie o sistema carregando: R0 ← 00h , R1 ← 01h , R2 ← 02h , R3 ← 03h ,
R4 ← 04h , R5 ← 05h , R6 ← 06h , R7 ← 07h

B. Reset a carry flag.

C. Faça uma leitura de uma porta de entrada (P1) e através da análise deste byte possibilite a escolha de uma das funções descritas na tabela a seguir.

D. Programa entra em loop e volta ao passo " C ".

| Bit | = 0 | = 1 | |
|---|---|---|---|
| P1.7 | ---------- | Envia p/ P2 o conteúdo de R0, R1,...R7<br>Separados por um "pequeno" delay | |
| P1.6 | ---------- | Incrementar o conteúdo de todos Rn | |
| P1.5 | ---------- | Decrementar o conteúdo de todos Rn | |
| P1.4 | ---------- | Rotacionar todos Rn de 4 bits ou<br>Trocar os seus respectivos nibbles | |
| P1.3 | ---------- | Rotacionar todos Rn de 3 bits | |
| P1.2 | ---------- | Rotacionar todos Rn de 2 bits | |
| P1.1 | ---------- | Rotacionar todos Rn de 1 bit | |
| P1.0 | Escolha do sentido de Rotação: p/ esquerda | Escolha do sentido de rotação: p/ direita | OBS: p/ os bits 4 ou 3 ou 2 ou 1 = "1" |

OBS:

- Admitir que o sistema nunca fornecerá 2 bits iguais a "1" simultaneamente nos pinos: P1.7, P1.6, P1.5, P1.4 P1.3, P1.2 e P1.1.
- Antes de iniciar a simulação fazer P1 = 00h (condição inicial).
- Simular "passo a passo" o programa observando sua execução.

SUGESTÕES:

- Usar as instruções: RL A ; RR A ; SWAP A ; CLR C ; JNB bit, rel ; LJMP end
- Executar cada uma das tarefas solicitadas em sub-rotinas especializadas e selecionadas pelo programa principal como mostrado no fragmento de programa a seguir:

```
        .
        .
        .
        JNB     P1.7, L1
        ACALL   SUB_7

L1:     JNB     P1.6, L2
        ACALL   SUB_6

L2:     JNB     P1.5, L3
        ACALL   SUB_5

L3:     JNB     P1.4, L4
        ACALL   SUB_4

L4:     JNB     P1.3, L5
        ACALL   SUB_3

L5:     JNB     P1.2, L6
        ACALL   SUB_2

L6:     JNB     P1.1, L7
        ACALL   SUB_1

L7:     .
        .
        .
```

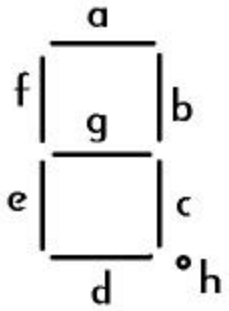
**EXERCÍCIO / EXEMPLO 6**

Escrever um programa que realize as seguintes operações:

A. Inicialmente leia o byte presente na porta P0.
B. Depois atualize o display ligado na porta P1 com o valor em hexadecimal correspondente. ao número binário formado pelos bits presentes nos pinos: P0.3 P0.2 P0.1 P0.0 .
C. O programa deve entrar em loop.

Características do display e da conexão:

- Display de LED's, catodo comum e de 7 segmentos + ponto decimal




- Ligação dos segmentos (através de um resistor de 220 ohms):

| pinos da porta 1: | P1.7 | P1.6 | P1.5 | P1.4 | P1.3 | P1.2 | P1.1 | P1.0 |
|---|---|---|---|---|---|---|---|---|
| segmentos: | h | g | f | e | d | c | b | a |

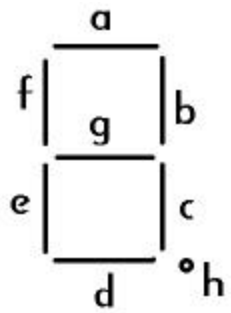
- Tabela de códigos de acendimento:

| Caractere | h P1.7 | g P1.6 | f P1.5 | e P1.4 | d P1.3 | c P1.2 | b P1.1 | a P1.0 | Código |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 3FH |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 06H |
| 2 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 5BH |
| 3 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 4FH |
| 4 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 66H |
| 5 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 6DH |
| 6 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 7DH |
| 7 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 07H |
| 8 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 7FH |
| 9 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 6FH |
| A | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 77H |
| B | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 7CH |
| C | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 39H |
| D | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 5EH |
| E | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 79H |
| F | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 71H |

```
;*********************************************************************
;          EXERCÍCIO / EXEMPLO 6 – DISPLAY & USO DE UMA TABELA
;*********************************************************************
                ORG     0000H
                LJMP    INICIO

        ;tabela de códigos de acendimento
                ORG     0040H
                DB      3FH                     ;código de acendimento do caractere "0"
                DB      06H                     ;código de acendimento do caractere "1"
                DB      5BH                     ;código de acendimento do caractere "2"
                DB      4FH                     ;código de acendimento do caractere "3"
                DB      66H                     ;código de acendimento do caractere "4"
                DB      6DH                     ;código de acendimento do caractere "5"
                DB      7DH                     ;código de acendimento do caractere "6"
                DB      07H                     ;código de acendimento do caractere "7"
                DB      7FH                     ;código de acendimento do caractere "8"
                DB      6FH                     ;código de acendimento do caractere "9"
                DB      77H                     ;código de acendimento do caractere "A"
                DB      7CH                     ;código de acendimento do caractere "B"
                DB      39H                     ;código de acendimento do caractere "C"
                DB      5EH                     ;código de acendimento do caractere "D"
                DB      79H                     ;código de acendimento do caractere "E"
                DB      71H                     ;código de acendimento do caractere "F"

                ORG     0060H
INICIO:         MOV     DPTR, #0040H            ;estabelece ponteiro no inicio da tabela
L1:             MOV     A, P0                   ;lê byte da porta P0
                ANL     A, #0FH                 ;mascara os 4 bits mais significativos
                MOVC    A, @A+DPTR              ;lê código de acendimento do caractere
                MOV     P1, A                   ;envia para o display
                AJMP    L1

                END
```

**EXERCÍCIO 7**

Alterar o programa anterior para que o número binário presente nos 4 bits mais significativos da porta P0 (P0.7 P0.6 P0.5 P0.4) seja mostrados (em hexadecimal) em um novo display conectado na porta P2, com as mesmas características em hardware do anterior.




- Ligação dos segmentos (através de um resistor de 220 ohms):

| pinos da porta 2: | P2.7 | P2.6 | P2.5 | P2.4 | P2.3 | P2.2 | P2.1 | P2.0 |
|---|---|---|---|---|---|---|---|---|
| segmentos: | h | g | f | e | d | c | b | a |

**EXERCÍCIO 8**

Escrever um programa que execute os seguintes passos:

A. Inicialmente desvie dos endereços reservados aos vetores de interrupção, (Interrupção externa número 0 = INT0' = pino P3.2 e Interrupção externa número 1 = INT1' = pino P3.3), para uma região de memória onde estará a rotina de tratamento de cada interrupção.
Exemplo:

```
        ORG     0000H
        LJMP    INICIO          ;Desvia dos vetores de interrupção

        ORG     0003H           ;Interrupção externa número 0
        LJMP    INT0

        ORG     0013H           ;Interrupção externa número 1
        LJMP    INT1

        ORG     0050H
INICIO: MOV     IE, #...
        .
        .
        .
```

B. Habilita e configura as interrupções externas através dos registradores "IE", "IP" e "TCON".
C. Posiciona SP (Stack pointer) em 0030H.
D. Envia 00H para as portas P0 e P1.
E. A cada interrupção externa incrementa o conteúdo da porta P0 (para o INT0) e o da porta P1 (para o INT1) criando dois contadores de interrupções.
F. Entra em loop.

OBS:

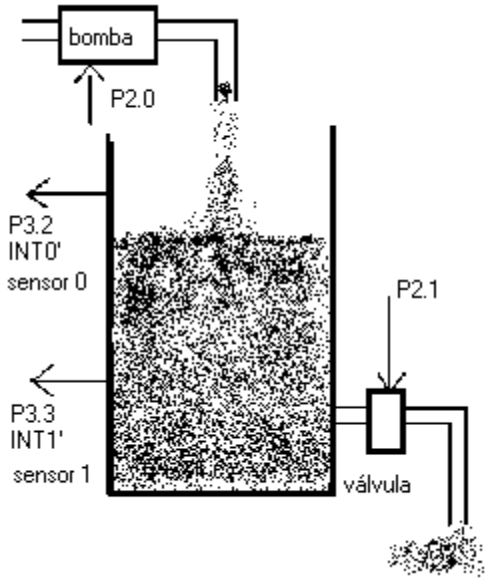
- Simular "passo a passo" o programa observando sua execução.
- Durante a simulação, controlar os bits de entrada de cada interrupção, verificando o funcionamento destas.

**EXERCÍCIO 9**

Escrever um programa para controlar o reservatório da figura a seguir, com o objetivo de manter o nível do líquido deste, que é c consumido pelo sistema, entre a posição dos sensores 0 e 1 (sensores 0 e 1), além de também gerar continuamente um sinal indicador de que o sistema está em operação em um LED, ligado no bit P1.7, alternado o status deste entre "0" e "1" , separado por um pequeno delay.

Definição das variáveis:

| | | | |
|---|---|---|---|
| sensor 0 = 1 | → líquido detectado | sensor 0 = 0 | → caso contrário |
| sensor 1 = 1 | → líquido detectado | sensor 1 = 0 | → caso contrário |
| bomba = 1 | → acionada | bomba = 0 | → desligada |
| válvula = 1 | → aberta | válvula = 0 | → fechada |




OBS:

- Apesar do sistema apresentado ser semelhante ao do exercício número 3 o programa deve ser diferente, pois nessa aplicação são usadas as duas interrupções externas no seu controle (ligação dos sensores).
- Para que o uso desse recurso seja possível é necessário que o programa salte os endereços de memória reservados para todas as interrupções (diretiva "ORG") e também configure aquelas usadas pelo sistema, através dos registradores "IE", "IP" e "TCON".
- O uso de interrupções é uma das características mais importantes em aplicações de controle envolvendo microcontroladores e microprocessadores, pois evita-se que a CPU fique monitorando continuamente os dispositivos de entrada. Assim o sistema proposto no exercício tem "liberdade" para ficar alternando o bit de saída da porta P1 (P1.7), entre "0" e "1", enquanto nenhuma interrupção é ativada.
- Para a resolução deste problema recomenda-se inicialmente a elaboração de um fluxograma ou algoritmo e após, a sua codificação em assembly, (notar que as rotinas de interrupção possuem fluxogramas separados do principal).
- O enunciado é propositadamente aberto, assim qualquer dado ou condição adicional pode ser adotada pelo programador.
- Após a simulação comparar esta solução com a empregada no exercício número 3.

**EXERCÍCIO / EXEMPLO 10**

- Dado o seguinte sistema microcontrolado baseado na CPU 8751:

  Motor de passo ligado nos bits: P1.3 , P1.2 , P1.1 e P1.0 conforme a tabela:

| Tabela de acionamento do motor | | | | |
|---|---|---|---|---|
| A→ P1.3 | B→ P1.2 | A'→ P1.1 | B'→ P1.0 | |
| 0 | 0 | 0 | 1 | Passo 1 |
| 0 | 0 | 1 | 0 | Passo 2 |
| 0 | 1 | 0 | 0 | Passo 3 |
| 1 | 0 | 0 | 0 | Passo 4 |

Escrever um programa que execute o controle do sistema da seguinte forma:

- Movimente o motor continuamente, com o sentido de rotação selecionado pelo bit de entrada P3.7:
  P3.7 = 0 → sentido horário
  P3.7 = 1 → sentido anti-horário

```
;*************************************************************************************************
;       EXERCÍCIO / EXEMPLO 10 – CONTROLE DE MOTOR DE PASSO
;       OBS:  Necessário o ajuste do delay em função da velocidade do microcomputador usado para a simulação.
;*************************************************************************************************
                ORG     0000H
                LJMP    INICIO

                ORG     0050H
INICIO:         MOV     A, #00010001B       ;estabelece o byte inicial
                                            ;para o chaveamento do motor
L2:             JNB     P3.7, L1            ;testa o bit de controle

                MOV     P1, A               ;movimenta no sentido anti-horário
                ACALL   DELAY               ;fornece uma pausa "entre passos"
                RR      A                   ;desloca um bit para a direita
                AJMP    L2                  ;retorna para novo teste

L1:             MOV     P1, A               ;movimenta no sentido horário
                ACALL   DELAY               ;fornece uma pausa "entre passos"
                RL      A                   ;desloca um bit para a esquerda
                AJMP    L2                  ;retorna para novo teste

DELAY:          MOV     R0, #1FH            ;pausa (ajustar se necessário)
L3:             DEC     R0
                CJNE    R0, #00H, L3
                RET

                END
```

**EXERCÍCIO 11**

Acrescentar ao sistema anterior uma chave para habilitar o movimento, ligada no pino P3.0 e operando da seguinte forma:

para P3.0 = 0 → motor parado
para P3.0 = 1 → motor em movimento (com sentido escolhido por P3.7).

**EXERCÍCIO 12**

Acrescentar ao sistema anterior um controle de posição angular (contador de passos) da seguinte forma:

- O número de passos (em binário) desejado para o movimento do motor é lido da porta P2 (até 255 passos ou 11111111b).
- Até o término do movimento escolhido (quando habilitado, com sentido de rotação selecionado e número de passos definido) o sistema ignora qualquer mudança nas variáveis de controle.
- Uma nova leitura das variáveis de controle deve acontecer somente após o final do movimento anterior.

**EXERCÍCIO 13**

Incluir no sistema anterior um controle de velocidade, possibilitando a escolha gradual desta através dos bits: P0.3, P0.2, P0.1 e P0.0 da seguinte forma:

- Bits para a seleção de velocidade dos dois motores:
  para P0.3 P0.2 P0.1 P0.0 = 1111 → temos a menor velocidade
  para P0.3 P0.2 P0.1 P0.0 = 0000 → temos a maior velocidade
  (Possibilitando 16 valores diferentes de velocidades).

**EXERCÍCIO 14**

Alterar o sistema / programa anterior para:

- Incluir um "push-button" de controle (sinal de emergência) ligado no pino de interrupção: P3.2 = INT0'
- Onde "push-button" pressionado equivale a "0" lógico.
- O motor deve ficar parado enquanto o "push-button" estiver pressionado e em movimento caso contrário.

**EXERCÍCIO 15**

Ampliar o sistema anterior incluindo um outro motor com o seu bit de controle de sentido de rotação e o sinal de parada de emergência ligado na interrupção, formando o seguinte conjunto:

- Bits de ligação do motor 1 e motor 2 conforme a tabela:

| Tabela de acionamento do motor 2 | | | | | Tabela de acionamento do motor 1 | | | | |
|---|---|---|---|---|---|---|---|---|---|
| A→ P1.7 | B→ P1.6 | A'→ P1.5 | B'→ P1.4 | | A→ P1.3 | B→ P1.2 | A'→ P1.1 | B'→ P1.0 | |
| 0 | 0 | 0 | 1 | Passo 1 | 0 | 0 | 0 | 1 | Passo 1 |
| 0 | 0 | 1 | 0 | Passo 2 | 0 | 0 | 1 | 0 | Passo 2 |
| 0 | 1 | 0 | 0 | Passo 3 | 0 | 1 | 0 | 0 | Passo 3 |
| 1 | 0 | 0 | 0 | Passo 4 | 1 | 0 | 0 | 0 | Passo 4 |

- Bits de controle do sentido de rotação dos motores:
  - P3.7 = 0 → motor 1 operando no sentido horário
  - P3.7 = 1 → motor 1 operando no sentido anti-horário
  - P3.6 = 0 → motor 2 operando no sentido horário
  - P3.6 = 1 → motor 2 operando no sentido anti-horário

- Bits para habilitação do movimento dos dois motores:
  - P3.1 P3.0 = 00 → motor 2 parado e motor 1 parado
  - P3.1 P3.0 = 01 → motor 2 parado e motor 1 em movimento
  - P3.1 P3.0 = 10 → motor 2 em movimento e motor 1 parado
  - P3.1 P3.0 = 11 → motor 2 em movimento e motor 1 em movimento

- Bits para a determinação do número passos dos dois motores:
  O número de passos de é lido da porta P2 .

- Bits para a seleção de velocidade dos dois motores:
  para P0.3 P0.2 P0.1 P0.0 = 1111 → temos a menor velocidade
  para P0.3 P0.2 P0.1 P0.0 = 0000 → temos a maior velocidade
  (Possibilitando 16 valores diferentes de velocidades).

- Sinais de emergência (interrupção) dos motores:
  P3.2 = INT0 para o motor 1 e 2

**EXERCÍCIO 16**

Ampliar o sistema anterior incluindo um "push-button" (sinal de emergência) específico para cada motor formando o seguinte conjunto:

- Sinais de emergência (interrupção) dos motores:
  P3.2 = INT0 para o motor 1
  P3.3 = INT1 para o motor 2

**EXERCÍCIO 17**

Alterar o sistema anterior para que os motores possam operar com velocidades diferentes, da seguinte forma:

- Bits para a seleção de velocidade do motor 1:
  para P0.3 P0.2 P0.1 P0.0 = 1111 → temos a menor velocidade
  para P0.3 P0.2 P0.1 P0.0 = 0000 → temos a maior velocidade
  (Possibilitando 16 valores diferentes de velocidades para o motor 1).

- Bits para a seleção de velocidade do motor 2:
  para P0.7 P0.6 P0.5 P0.4 = 1111 → temos a menor velocidade
  para P0.7 P0.6 P0.5 P0.4 = 0000 → temos a maior velocidade
  (Possibilitando 16 valores diferentes de velocidades para o motor 2).

**EXERCÍCIO 18**

- Dado o seguinte sistema microcontrolado baseado na CPU 8751:
  Motor de passo ligado nos bits: P1.3 , P1.2 , P1.1 e P1.0 conforme a tabela:

| Tabela de acionamento do motor | | | | |
|---|---|---|---|---|
| A→ P1.3 | B→ P1.2 | A'→ P1.1 | B'→ P1.0 | |
| 0 | 0 | 0 | 1 | Passo 1 |
| 0 | 0 | 1 | 0 | Passo 2 |
| 0 | 1 | 0 | 0 | Passo 3 |
| 1 | 0 | 0 | 0 | Passo 4 |

Escrever um programa que execute o controle a aceleração do motor da seguinte forma:

- Movimente o motor continuamente, com a habilitação do movimento dada pelo bit P3.0 e o sentido deste pelo bit P3.7 como descrito a seguir:
  P3.7 = 0 → movimento no sentido horário
  P3.7 = 1 → movimento no sentido anti-horário
  Para P3.0 = 0 → motor parado
  Para P3.0 = 1 → motor em movimento (com sentido escolhido por P3.7).

- Se habilitado, inicie seu movimento em baixa velocidade e gradualmente acelere até estabilizar em uma velocidade adotada como máxima.

**EXERCÍCIO 19**

Alterar o sistema anterior para que o motor realize automaticamente o seguinte ciclo de trabalho:

A. Se habilitado, inicie seu movimento em baixa velocidade e gradualmente acelere até estabilizar em uma velocidade adotada como máxima.
B. Após um determinado tempo operando nesta situação, diminua gradualmente a velocidade até um valor mínimo.
C. A leitura dos bits de controle de sentido de rotação e habilitação, será feita sempre quando o sistema atingir a menor velocidade, definindo assim o próximo movimento.
D. O sistema entra em loop.

**EXERCÍCIO 20**

Incluir no sistema anterior um controle adicional de aceleração, feito pelo bit P3.6, da seguinte forma:

P3.6 = 0 → o motor inicia seu movimento em baixa velocidade e gradualmente acelere até estabilizar em uma velocidade adotada como máxima.

P3.6 = 1 → o motor inicia seu movimento em um velocidade adotada como máxima e gradualmente diminua esta, até estabilizar em uma velocidade baixa.

**EXERCÍCIO 21**

Alterar o sistema / programa anterior para:

- Incluir um "push-button" de controle (sinal de emergência) ligado no pino de interrupção: P3.2 = INT0'
- Onde "push-button" pressionado equivale a "0" lógico.
- O motor deve ficar parado enquanto o "push-button" estiver pressionado e em movimento caso contrário.

**EXERCÍCIO / EXEMPLO 22**

Escrever um programa para automatizar um sistema constituído por um estacionamento de automóveis hipotético com as características abaixo:

ENTRADAS DO SISTEMA:

| porta | Descrição: | bits / pinos | níveis lógicos | OBS: |
|---|---|---|---|---|
| P3 | Push-button para iniciar a operação do sistema (botão de START). | P3.0 | 0 = para iniciar<br>1 = caso contrário | Acionado uma única vez no período. |
| | Sinal de indicador de falta de energia elétrica. | P3.2 = INT0' | 0 = falta de energia<br>1 = caso contrário | Fornecido pelo gerador. |
| | Sinal de contagem de veículos que entram no estacionamento. | P3.4 = T/C0 | transição de 1 p/ 0 indica entrada de um veículo | Fornecido pelo sistema de entrada. |

SAÍDAS DO SISTEMA:

| porta | Descrição: | bits / pinos | níveis lógicos | OBS: |
|---|---|---|---|---|
| P0 | Controle de painel luminoso de propaganda do estacionamento, com 8 lâmpadas controladas individualmente e seqüencialmente. | P0.7, P0.6, P0.5, P0.4, P0.3, P0.2, P0.1 e P0.0 | 0 = apagada<br>1 = acesa | Lâmpadas piscando durante toda a operação do sistema. |
| P1 | Controle da portaria de entrada (cancela). | P1.7 | 0 = abre<br>1 = fecha | Abre no início e fecha quando as vagas estão esgotadas. Considerar o tempo necessário para o movimento da cancela. |
| | Luz indicadora de estacionamento em funcionamento. | P1.6 | 0 = apagada<br>1 = acesa | Quando acesa indica estacionamento funcionando e apagada, estacionamento fechado. |
| | Controle da iluminação de emergência. | P1.0 | 0 = apagada<br>1 = acesa | |

OBS:

- Neste exemplo só é considerado o total de automóveis que entram no estacionamento no período de funcionamento e para facilitar a simulação é adotada uma capacidade máxima de 10 veículos. Quando o número de vagas for esgotado o sistema para e só inicia novamente o seu funcionamento através do acionamento do botão START.
- Na simulação, atuar nas variáveis de entrada do sistema (bits P3.0, P3.2 e P3.4) e observar que, durante a contagem dos automóveis que entram no estacionamento (conteúdo de TL0 sendo alterado) e na falta de energia elétrica, a porta P0 continua controlando o painel luminoso e a porta P1 atuando em outras saídas de controle.
- Para a resolução de um projeto desse porte, é recomendável a utilização de tabelas como as anteriores, onde são descritas as funções das variáveis, os pinos de ligação e a correspondência dos níveis lógicos.

```
;*************************************************************************
;       EXERCÍCIO / EXEMPLO 22  -  AUTOMAÇÃO DE ESTACIONAMENTO
;*************************************************************************
                ORG             0000H
                LJMP            INICIO

                ORG             0003H
                LJMP            INT0

                ORG             000BH
```

```
                LJMP        CNT0

                ORG         0050H
INICIO:         MOV         P0, #00H            ;apaga painel luminoso
                CLR         P1.0                ;apaga iluminação de emergência
                SETB        P1.7                ;fecha cancela de entrada
                CLR         P1.6                ;apaga luz indicadora de funcionamento
                ACALL       DELAY_2             ;delay para o movimento mecânico
L1:             JB          P3.0, L1            ;espera pressionar START
L2:             JNB         P3.0, L2            ;espera soltar  START
                MOV         TL0, #0F6H          ;estabelece o valor inicial do contador
                MOV         TH0, #0F6H          ;calculo: 255 – 10 + 1 = 246 = F6H
                MOV         A, #00000001B       ;estabelece o byte inicial do painel
                MOV         IE, #10000011B      ;habilita INT0 e T/C0
                MOV         IP, #00000001B      ;prioridades: INT0 alta  e T/C0 baixa
                MOV         TMOD, #00000110B    ;define modo 2 para T/C0
                MOV         TCON, #00010001B    ;INT0 ativa por  rampa e liga T/C0
                SETB        P1.6                ;acende luz indicadora  funcionamento
                CLR         P1.7                ;abre cancela de entrada
                ACALL       DELAY_2             ;delay para movimento mecânico
L3:             JNB         P3.2, L4            ;desvia se energia elétrica OK
                CLR         P1.0                ;apaga iluminação de emergência
L4:             MOV         P0, A               ;controle do painel luminoso
                RL          A                   ;desloca o bit da lâmpada acesa
                ACALL       DELAY_1             ;delay para as lâmpadas
                AJMP        L3

INT0:           SETB        P1.0                ;acende iluminação de emergência
                RETI                            ;retorna da interrupção

CNT0:           SETB        P1.7                ;fecha cancela de entrada
                ACALL       DELAY_2             ;delay para movimento mecânico
                CLR         P1.6                ;apaga luz de funcionamento
                LJMP        L1                  ;volta para aguardar o START
                RETI                            ;esta instrução nunca é executada

DELAY_1:        MOV         R7, #0AH            ;fornece delay para as lâmpadas
L5:             DEC         R7
                CJNE        R7, #00H, L5
                RET

DELAY_2:        MOV         R6, #50H            ;fornece  delay para movimento
L6:             DEC         R6                  ;mecânico
                CJNE        R6, #00H, L6
                RET

                END
```

**EXERCÍCIO 23**

Modificar o sistema anterior incluindo outros sinais de entrada (sensor para os automóveis) e de saída (controle de uma cancela de saída), para possibilitar que este considere os automóveis que saem do estacionamento, liberando vagas durante o período de funcionamento.

ENTRADAS ADICIONAIS AO SISTEMA:

| porta | Descrição: | bits / pinos | níveis lógicos | OBS: |
|---|---|---|---|---|
| P3 | Sinal indicador de saída de veículo | P3.6 | 0 = saída de veículo<br>1 = caso contrário | |

SAÍDAS ADICIONAIS AO SISTEMA:

| porta | Descrição: | bits / pinos | níveis lógicos | OBS: |
|---|---|---|---|---|
| P1 | Controle da portaria de saída (cancela). | P1.5 | 0 = abre<br>1 = fecha | Fecha no início e abre quando é detectado um veículo saindo. Considerar o tempo necessário para o movimento da cancela. |

**EXERCÍCIO 24**

Sofisticar o sistema anterior, incluindo um contador para controlar o número de funcionários do estacionamento presentes durante o período de funcionamento.

ENTRADAS ADICIONAIS AO SISTEMA:

| porta | Descrição: | bits / pinos | níveis lógicos | OBS: |
|---|---|---|---|---|
| P3 | Sinal de contagem dos funcionários que chegam ao trabalho | P3.5 = T/C1 | Transição de 1 p/ 0 indica chegada do funcionário | Fornecido pelo sistema de entrada |

**EXERCÍCIO 25**

Incluir no sistema anterior o controle de uma lâmpada indicadora de que o número de funcionários presentes no estacionamento é suficiente para o funcionamento do mesmo.

OBS:

- Por razões de segurança, o estacionamento só entrará em funcionamento quando o número de funcionários presentes for maior ou igual a 5.

SAÍDAS ADICIONAIS AO SISTEMA:

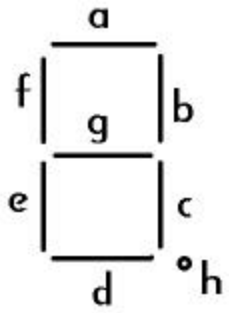
| porta | Descrição: | bits / pinos | níveis lógicos | OBS: |
|---|---|---|---|---|
| P1 | Controle da lâmpada indicadora de número de funcionários suficiente. | P1.4 | 0 = apagada<br>1 = acesa | Acende quando o número de funcionários é suficiente. |

**EXERCÍCIO 26**

Incluir no sistema a utilização da porta P2 como saída, com um display de 7 segmentos conectado (anodo comum), que sempre indica o número de vagas disponíveis naquele instante no estacionamento (em decimal) da seguinte forma: para nenhuma vaga disponível manter o display apagado e para 10 vagas acender o número zero.

OBS:

- Tomar como referência a solução do exercício 6




- Ligação dos segmentos (através de um resistor de 220 ohms):

| pinos da porta 2: | P2.7 | P2.6 | P2.5 | P2.4 | P2.3 | P2.2 | P2.1 | P2.0 |
|---|---|---|---|---|---|---|---|---|
| segmentos: | h | g | f | e | d | c | b | a |

**EXERCÍCIO / EXEMPLO 27**

Projetar um sistema de controle de um elevador para 3 níveis, baseado no microcontrolador 8751, com as seguintes características:

Entradas do sistema: P0 e INT0

| Sinal | Descrição | Pino de ligação | Níveis lógicos |
|---|---|---|---|
| CH1 | Botão para enviar o elevador ao 1º andar (1º nível ou térreo) | P0.0 | 1=acionada<br>0=caso contrário |
| CH2 | Botão para enviar o elevador ao 2º andar | P0.1 | 1=acionada<br>0=caso contrário |
| CH3 | Botão para enviar o elevador ao 3º andar | P0.2 | 1=acionada<br>0=caso contrário |
| Sensor1 | Sensor indicador de presença da cabine no 1º andar | P0.4 | 1= há cabine no andar<br>0=caso contrário |
| Sensor2 | Sensor indicador de presença da cabine no 2º andar | P0.5 | 1= há cabine no andar<br>0=caso contrário |
| Sensor3 | Sensor indicador de presença da cabine no 3º andar | P0.6 | 1= há cabine no andar<br>0=caso contrário |
| SAFETY | Botão de emergência | P3.2 = INT0' | 0=acionado<br>1=caso contrário |

Saídas do sistema: P1

| Sinal | Descrição | Pino de ligação | Níveis lógicos |
|---|---|---|---|
| Motor do elevador | Bits usados para a movimentação do motor | P1.1 P1.0 | P1.1 P1.0<br>0 0 elevador parado<br>0 1 elevador desce<br>1 0 elevador sobe<br>1 1 não usado |
| Porta | Bit usado para a movimentação da porta | P1.2 | 0=fecha porta<br>1=abre porta |
| LED1 | Led para indicar que a cabine está no 1º andar | P1.4 | 0=led apagado<br>1=led aceso |
| LED2 | Led para indicar que a cabine está no 2º andar | P1.5 | 0=led apagado<br>1=led aceso |
| LED3 | Led para indicar que a cabine está no 3º andar | P1.6 | 0=led apagado<br>1=led aceso |

OBS:

- Se desejável é possível associar-se o 1º andar da tabela anterior ao andar térreo de um edifício, deslocando-se também os outros níveis, sem alterações no funcionamento da solução fornecida como exemplo a seguir.
- Assim que o sistema é iniciado o elevador desloca-se para o primeiro andar com a porta fechada e lá chegando abre a porta.
- O elevador somente desloca-se com a porta fechada.

- Quando a cabine está parada, sua porta conecta-se mecanicamente com a porta do respectivo andar e assim ambas terão o mesmo movimento.
- Os botões de comando do elevador estão no interior da cabine, não existindo botões externos ao lado da porta de entrada, assim o usuário só pode entrar no elevador se este estiver parado no andar, sendo o destino escolhido pelo mesmo dentro da cabine (limitação inicialmente adotada para a simplificação do exemplo).
- O LED aceso indica a presença da cabine no respectivo andar.
- Normalmente não é recomendável uma freqüência de "clock" inferior a 3MHz para aplicações reais, más nesse problema, onde um dos objetivos é a determinação de um intervalo de tempo e a visualização do funcionamento de um dos TEMPORIZADORES através do simulador, esta freqüência fica estabelecida em 120KHz. (valor escolhido em função da aplicação).
- O tempo gasto pela porta no movimento de abrir e fechar deve ser de 5 segundos e determinado pelo TIMER0 (vide descrição do cálculo no cabeçalho do programa).

```
;*****************************************************************************************************
;       EXERCÍCIO / EXEMPLO 27:          CONTROLE DO ELEVADOR          05/11/2002      02:45h
;       freqüência do timer = (freqüência do clock) / 12 = 120K / 12 = 10K Hz e o seu período = 0,1x10^-3 segundos
;       assim para um tempo de 5 segundos temos: 5 / 0,1x10^-3 = valor da contagem = 50000
;       65535 – 50000 + 1 = 15536 = 3CB0H = valor a ser carregado inicialmente no contador / temporizador
;*****************************************************************************************************
                ORG             0000H
                LJMP            INICIO

                ORG             0003H
                LJMP            INT0

                ORG             000BH
                LJMP            TIMER0

                ORG             0050H
INICIO:         MOV             SP, #0030H          ;estabelece o ponteiro da pilha
                MOV             IE, #10000011B      ;habilita INT0 e TIMER0
                MOV             IP, #00000001B      ;define prioridade alta p/ INT0 e baixa p/ TIMER0
                MOV             TCON, #00000000B    ;INT0 ativa por nível "0" e TIMER0 desligado
                MOV             TMOD, #00000001B    ;função de temporização com controle do TIMER0 feito pelo bit TR0
                                                    ;modo 1 de operação
                CLR             P1.1                ;para o elevador
                CLR             P1.0
                MOV             TH0, #3CH           ;valor inicial para a contagem / temporização
                MOV             TL0, #0B0H
                CLR             00                  ;inicializa um bit endereçável para ser a FLAG DO TIMER0
;*****************************************************************************************************
;O sistema (elevador) está sendo iniciado
;*****************************************************************************************************
                JB              P0.4, L25           ;verifica se o elevador já estava no 1º andar.
                ACALL           DESCER              ;leva o elevador inicialmente para o térreo.
L1:             JNB             P0.4, L1            ;espera chegar no 1º andar
                ACALL           PARAR
L25:            SETB            P1.4                ;acende led indicador de 1º andar
                CLR             P1.5                ;apaga led indidador de 2º andar
                CLR             P1.6                ;apaga led indicador de 3º andar
;*****************************************************************************************************
;O elevador está no 1º andar
;*****************************************************************************************************
ANDAR_1:        JB              P0.1, L2            ;testa chamada para o 2º andar
                JB              P0.2, L3            ;testa chamada para o 3º andar
                AJMP            ANDAR_1             ;volta para esperar uma chamada
L2:             ACALL           SUBIR               ;movimenta o elevador para o 2º andar
L4:             JNB             P0.5, L4            ;espera chegar no 2º andar
```

```
                ACALL       PARAR
                SETB        P1.5            ;acende o led indicador de 2º andar
                LJMP        ANDAR_2
L3:             ACALL       SUBIR           ;movimenta o elevador para o 3º andar
L5:             JNB         P0.6, L5        ;espera chegar no 3º andar
                ACALL       PARAR
                SETB        P1.6            ;acende o led indicador de 3º andar
                LJMP        ANDAR_3
;*************************************************************************************************
;O elevador está no 2º andar
;*************************************************************************************************
ANDAR_2:        JB          P0.0, L6        ;testa chamada para o 1º andar
                JB          P0.2, L7        ;testa chamada para o 3º andar
                AJMP        ANDAR_2         ;volta para esperar uma chamada
L6:             ACALL       DESCER          ;movimenta o elevador para o 1º andar
L8:             JNB         P0.4, L8        ;espera chegar no 1º andar
                ACALL       PARAR
                SETB        P1.4            ;acende o led indicador de 1º andar
                LJMP        ANDAR_1
L7:             ACALL       SUBIR           ;movimenta o elevador para o 3º andar
L9:             JNB         P0.6, L9        ;espera chegar no 3º andar
                ACALL       PARAR
                SETB        P1.6            ;acende o led indicador de 3º andar
                LJMP        ANDAR_3
;*************************************************************************************************
;O elevador está no 3º andar
;*************************************************************************************************
ANDAR_3:        JB          P0.0, L10       ;testa chamada para o 1º andar
                JB          P0.1, L11       ;testa chamada para o 2º andar
                AJMP        ANDAR_3         ;volta para esperar uma chamada
L10:            ACALL       DESCER          ;movimenta o elevador para o 1º andar
L12:            JNB         P0.4, L12       ;espera chegar no 1º andar
                ACALL       PARAR
                SETB        P1.4            ;acende o led indicador de 1º andar
                LJMP        ANDAR_1
L11:            ACALL       DESCER          ;movimenta o elevador para o 2º andar
L13:            JNB         P0.5, L13       ;espera chegar no 2º andar
                ACALL       PARAR
                SETB        P1.5            ;acende o led indicador de 2º andar
                LJMP        ANDAR_2
;*************************************************************************************************
;               sub-rotina:     SUBIR
;*************************************************************************************************
SUBIR:          CLR         P1.4            ;apaga todos os led's
                CLR         P1.5
                CLR         P1.6
                CLR         P1.2            ;fecha a porta
                SETB        TR0             ;liga o timer0
L20:            JNB         00, L20         ;espera o término da temporização
                CLR         00              ;reseta a FLAG DO TIMER0
                SETB        P1.1            ;sobe o elevador
                CLR         P1.0
                RET
;*************************************************************************************************
;               sub-rotina:     DESCER
;*************************************************************************************************
DESCER:         CLR         P1.4            ;apaga todos os led's
                CLR         P1.5
                CLR         P1.6
                CLR         P1.2            ;fecha a porta
```

```
                SETB    TR0             ;liga o timer0
L21:            JNB     00, L21         ;espera o término da temporização
                CLR     00              ;reseta a FLAG DO TIMER0
                CLR     P1.1            ;desce o elevador
                SETB    P1.0
                RET
;*****************************************************************************************************
;               sub-rotina:     PARAR
;*****************************************************************************************************
PARAR:          CLR     P1.1            ;para o elevador
                CLR     P1.0
                SETB    P1.2            ;abre a porta
                SETB    TR0             ;liga o timer0
L22:            JNB     00, L22         ;espera o término da temporização
                CLR     00              ;reseta a FLAG DO TIMER0
                RET
TIMER0:         CLR     TR0             ;desliga o timer0
                MOV     TH0, #3CH       ;valor inicial para a contagem / temporização
                MOV     TL0, #0B0H
                SETB    00              ;seta a FLAG DO TIMER0
                RETI                    ;retorna para o programa
INT0:           CLR     P1.1            ;para o elevador
                CLR     P1.0
                SETB    P1.2            ;abre a porta
L23:            JNB     P3.2, L23       ;espera o término da situação de emergência
                RETI
                END
```

**EXERCÍCIO 28:**

Ampliar o sistema anterior para que este seja aplicado em um edifício com 4 níveis, adicionando os seguintes recursos:

Entradas do sistema: P0

| Sinal | Descrição | Pino de ligação | Níveis lógicos |
|---|---|---|---|
| CH4 | Botão para enviar o elevador ao 4º andar | P0.3 | 1=acionada<br>0=caso contrário |
| Sensor4 | Sensor indicador de presença da cabine no 4º andar | P0.7 | 1= há cabine no andar<br>0=caso contrário |

Saídas do sistema: P1

| Sinal | Descrição | Pino de ligação | Níveis lógicos |
|---|---|---|---|
| LED4 | Led para indicar que a cabine está no 4o andar | P1.7 | 0=led apagado<br>1=led aceso |

**EXERCÍCIO 29:**

Melhorar a função da "rotina de tratamento da interrupção INT0" (rotina de emergência) do sistema anterior, para que ao ser acionada no momento em que o elevador estiver parado em um andar, abra a porta da cabine e para a situação do elevador estar entre andares, leve o mesmo ao andar imediatamente inferior antes de abrir sua porta. Incluir no sistema o recurso de um alarme sonoro, que deve ser disparado caso o elevador demore mais que um tempo determinado para chegar no andar inferior, após o acionamento deste botão de emergência, onde:

Saídas do sistema: P1

| Sinal | Descrição | Pino de ligação | Níveis lógicos |
|---|---|---|---|
| ALARME | Alarme sonoro | P1.3 | 0=alarme desligado<br>1=alarme ligado |

**EXERCÍCIO 30:**

Incluir no sistema anterior um teclado fora da cabine e em cada andar do edifício, com o objetivo de ampliar o controle do elevador, ligado na porta P2 do microcontrolador da seguinte forma:

Entradas do sistema: P0

| Sinal | Descrição | Pino de ligação | Níveis lógicos |
|---|---|---|---|
| CH1-up | Botão no 1º andar, para chamar o elevador para andares superiores. | P2.0 | 1=acionada<br>0=caso contrário |
| CH2-down | Botão no 2º andar, para chamar o elevador para andares inferiores. | P2.1 | 1=acionada<br>0=caso contrário |
| CH2-up | Botão no 2º andar, para chamar o elevador para andares superiores. | P2.2 | 1=acionada<br>0=caso contrário |
| CH3-down | Botão no 3º andar, para chamar o elevador para andares inferiores. | P2.3 | 1=acionada<br>0=caso contrário |
| CH3-up | Botão no 3º andar, para chamar o elevador para andares superiores. | P2.4 | 1=acionada<br>0=caso contrário |
| CH4-down | Botão no 4º andar, para chamar o elevador para andares inferiores. | P2.5 | 1=acionada<br>0=caso contrário |

OBS:

- Durante um movimento, o elevador somente deve atender um pedido de chamada, se a cabine estiver no mesmo andar onde foi feito essa chamada ou o chamado ocorrer em um andar que ainda irá ser transposto durante o seu atual movimento.

**EXERCÍCIO 31:**

Incluir no sistema anterior o recurso de enviar o elevador para o 1º andar e após abrir a sua porta, se não houver nenhuma chamada durante um tempo determinado.

# CAPÍTULO 4: PROJETOS

## 4.1 RECOMENDAÇÕES SOBRE PROJETOS E MONTAGENS DE SISTEMAS MICROCONTROLADOS

**ALIMENTAÇÃO:**

- Deve ser externa a placa.
- Conectar um capacitor de poliester (aproximadamente 10nF) ligado entre os pinos "Vcc" e "GND" , ao lado do microcontrolador.
- Conectar um capacitor eletrolitico (mínimo 10uF) na entrada da alimentação na placa.

**MONTAGEM:**

- Microcontrolador deve ficar em um soquete de pinos torneados.
- Usar conectores para os bits das portas envolvidos na aplicação (macho tipo "prego") ou fios soldados diretamente na placa..
- Usar bornes simples para a alimentação ou fios soldados diretamente na placa.
- Usar pés de borracha ou outro isolador, para a sustentação da placa.
- Usar uma placa padrão preferencialmente com as trilhas desenhadas como um "proto board" (evitar as placas com ilhas isoladas que são ideais para montagens em wire-wrap).

**OSCILADOR:**

- A família 8051 trabalha tipicamente com cristais de 8, 10, 12MHz, conforme o tipo da CPU e de no mínimo 3,5MHz (um valor fácil de ser encontrado é de 11,0592MHz, usado pelo sinal de croma de televisão).
- O encapsulamento metálico do cristal deve ser soldado ao GND da placa.
- As ligações do circuito oscilador devem ser as mais curtas possíveis.

**RESET:**

- Para o reconhecimento da CPU este sinal deve permanecer em "1" por 2 ou mais ciclos de máquina (clock na freqüência do cristal usado).
- Após o Reset:
    - O "PC", o "Acumulador", o registrador "B", os "Flags", o "DPTR" e os registros dos "TIMERS' são zerados.
    - O "SBUF" (buffer serial) estará com o conteúdo indeterminado e o "SCON" (serial control) será zerado.
    - Os registradores de controle de interrupção "IE" (Interrupt Enable) e "IP" (Interrupt Priority) terão o valor binário XXX00000.
    - No "SP" (stack pointer) é carregado o valor 07H.
    - As portas P0, P1, P2 e P3 terão o valor FFH.
- Durante o Reset:
    - O nível lógico dos pinos das portas é indeterminado, indo para "1" após esse período.
    - Deve-se considerar essas características para evitar o acionamento não desejado de qualquer periférico externo.
- A RAM interna não é afetada pelo "Reset forçado" ou partida quente (push-botton de Reset) e após o "Power-On" ou partida fria, o seu conteúdo é aleatório.

**PORTAS:**

- P0: Porta bidirecional de 8 bits.
  Cada pino desta porta pode suprir / drenar 2 cargas TTL.
  Quando configurada como entrada, terá o nível lógico de seus pinos flutuando na ausência de sinal.

- P1: Porta bidirecional de 8 bits com pull-ups internos.
  Cada pino desta porta pode suprir / drenar uma carga TTL ou várias CMOS sem pull-ups externos.
  Seus pinos tem sempre um estado definido ("1"ou "0"), possibilitando que o nível lógico do pino possa ser medido, mesmo quando usado como entrada.

- P2: Porta bidirecional de 8 bits com pull-ups internos.
  Cada pino desta porta pode suprir / drenar uma carga TTL ou várias CMOS sem pull-ups externos.
  Seus pinos tem sempre um estado definido ("1"ou "0"), possibilitando que o nível lógico do pino possa ser medido, mesmo quando usado como entrada.

- P3: Porta bidirecional de 8 bits com pull-ups internos, servindo também para funções especiais.
  Seus pinos tem sempre um estado definido (“1”ou “0”), possibilitando que o nível lógico do pino possa ser medido, mesmo quando usado como entrada.
  Com o uso alguma das funções especiais, a P3 deve ser tratada apenas como porta de I/O de apenas bit endereçável.

**HARDWARE ADICIONAL:**

- Usar LED’s indicadores de status do sistema, ativos em “0”, conforme esquema.
- Não existe a necessidade do uso de foto-acopladores na aplicação.

**SOFTWARE:**

- Para que o programa ocupe apenas a memória ROM interna deve-se manter o PC < 4096 ou PC < 0FFFH, assim otimize esse espaço para alojar o seu programa e cuidado com aplicações que necessitem de grandes tabelas.
- Inicie o desenvolvimento do programa com um algoritmo ou fluxograma correspondente, tão detalhado quanto for a sua necessidade de compreender todos os passos intermediários necessários na aplicação.
- Durante o desenvolvimento, criar versões intermediárias do software para testar o hardware e rotinas importantes do projeto, jamais deixe para executar os testes na última versão do programa.

**ESCOLHA DA CPU:**

- Recomenda-se o uso de uma das seguintes CPU’s:
  - 8751 da intel (modelo com EPROM)
  - AT89S51, AT89S52, AT89S53 ou AT89S8252 da ATMEL (modelos com memória FLASH)
- A escolha de uma destas deve levar em conta qual o gravador disponível na instituição.

**PROGRAMAÇÃO:**

- Para modelos com EPROM:
  - O grupo deve usar o “apagador” de forma segura não expondo-se a radiação ultra-violeta (o tempo de apagamento é de aproximadamente 20 min e tende a aumentar com o número de operações).
  - O grupo deve procurar o Professor para receber orientação o sobre correto uso do gravador.
  - O arquivo “.HEX” pode ser usado para gravação do programa no microcontrolador de duas formas:
    - Usando o software de comunicação PC ←→ Gravador (gravação automática).
    - Digitando-se diretamente o arquivo em hexadecimal no gravador (gravação manual), sendo que para isso deve-se imprimir o arquivo “.HEX” (formato Hexa Intel) e separar os bytes do programa (que serão digitados) daqueles usados apenas para a comunicação (que devem ser rejeitados) conforme exemplo e descrições abaixo:

      Neste formato temos a seguinte composição:
      (Quantidade em Hexa) (Endereço do primeiro byte) 00 (Bytes do programa) CS
      Exemplo :10 00 20 00 **23 80 F5 03 80 F2 78 0E 75 8D 07 75 8B 53 D2 8E** 81
      Neste arquivo temos:
      - Os dois pontos (:);
      - 10, indicando que são 10H (16 em decimal) os bytes na linha;
      - 0020, indicando que o endereço da EPROM, do primeiro byte de programa na linha é 0020H;
      - 00 é a separação dos campos (sempre será 00H);
      - os 16 bytes da linha impressa (23 80 F5 03 80 F2 78 0E 75 8D 07 75 8B 53 D2 8E) os únicos que devem ser digitados;
      - CS é um byte de Check Sum, usado para a verificação de integridade do arquivo após a transmissão.
      - Assim é possível a gravação do arquivo sem a comunicação PC ←→ gravador, utilizando-se apenas os bytes referentes ao programa, observando-se o endereço inicial.
      - O arquivo “.LST” também pode ser usado para a gravação manual do programa, dispensando a retirada de bytes como no “.HEX” más provavelmente necessitando de mais folhas de papel para a impressão.
- Para modelos com memória FLASH:
  - Usando um simples circuito gravador (com esquema fornecido a seguir), que pode opcionalmente ser montado pelo grupo e um software espeçífico de comunicação PC ←→ Gravador.

**PRODUTO FINAL**

- Cpu montada em placa padrão, conforme esquema do sistema mínimo mostrado a seguir.
- Hardware adicional, necessário para a aplicação escolhida com forma de montagem livre.
- Documentação técnica incluindo:
    - Esquema eletrônico do hardware adicional.
    - Fluxograma ou algoritmo geral e listagem do software de controle comentada (arquivo .LST).

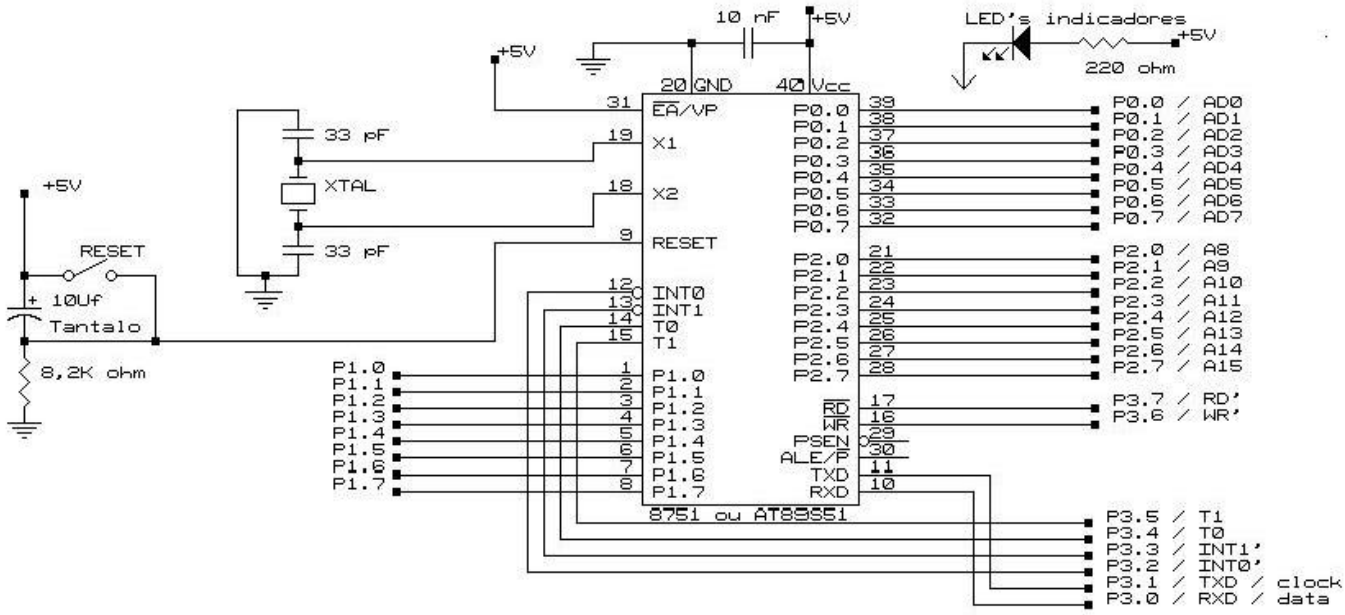
**AVALIAÇÃO**

- Trabalho em grupo de no máximo 3 ou 4 alunos (dependendo do Professor).
- Avaliação individual considerando-se o
    - O funcionamento do projeto.
    - A documentação entregue.
    - A efetiva participação de cada componente do grupo
    - Argüição feita pelo Professor individualmente aos alunos na data de apresentação.

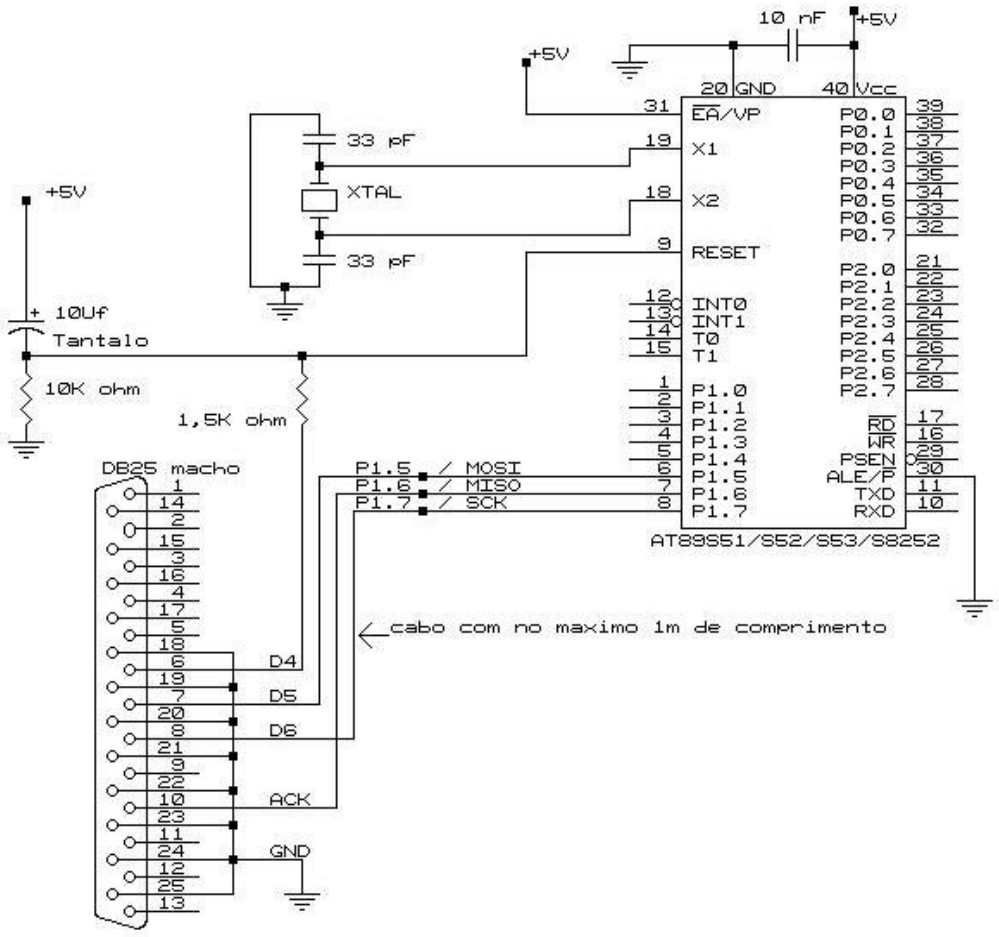
**CALENDÁRIO**

- Cadastro do grupo e escolha do tema do projeto:_____
- Datas para o desenvolvimento do projeto:_____
- Data limite para apresentação do projeto:_____

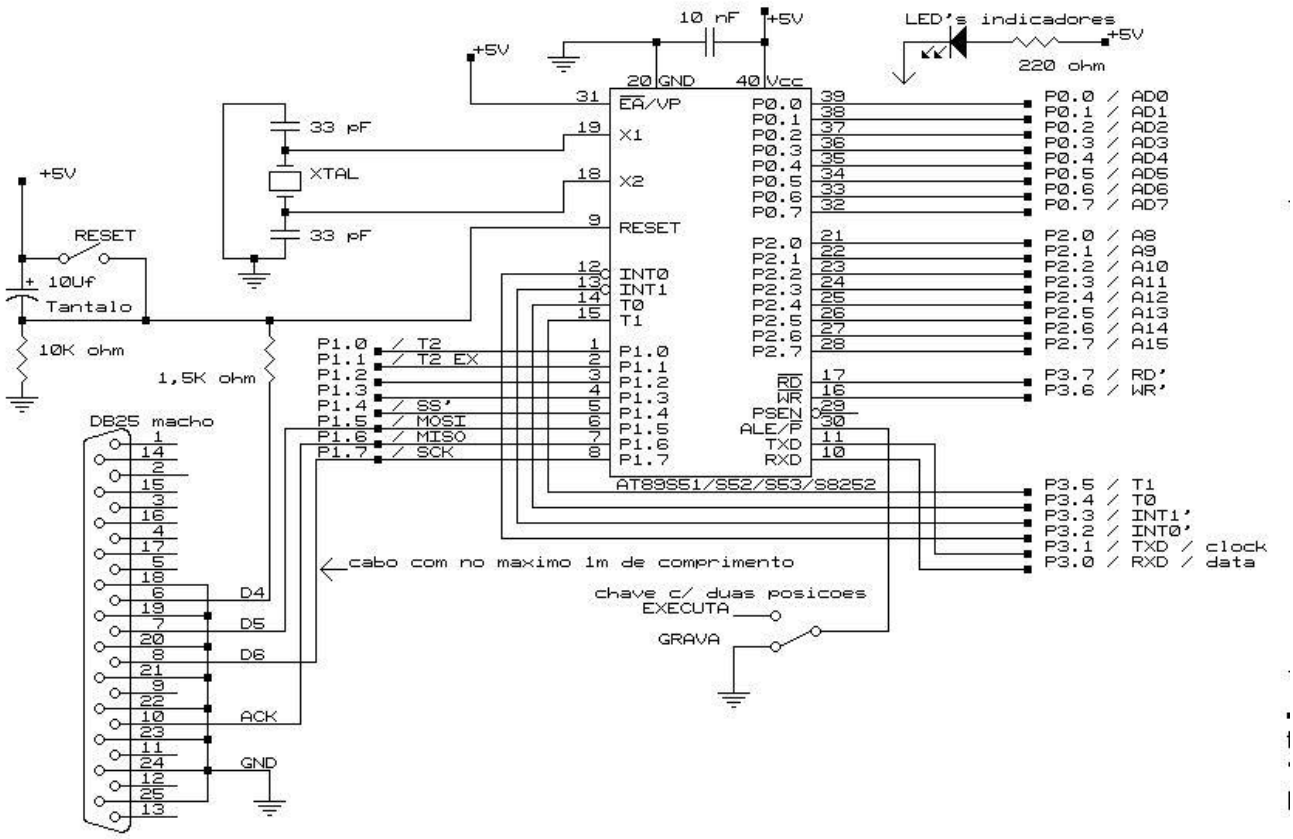
## 4.2 CPU PARA UM SISTEMA MÍNIMO



## 4.3 CIRCUITO DO GRAVADOR
**(para as cpu's AT89S51, AT89S52, AT89S53 e AT89S8252)**

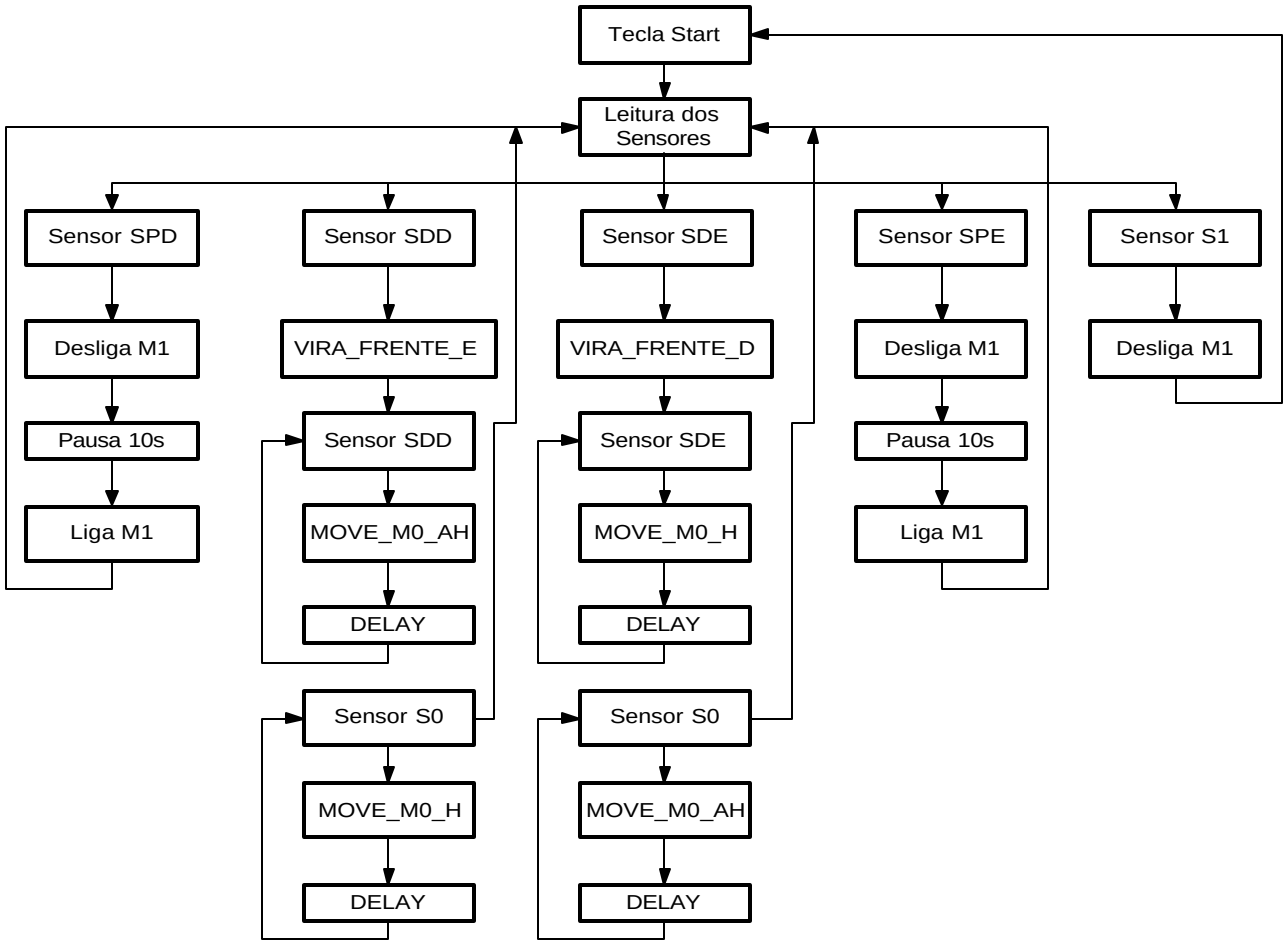


## 4.4 CPU DO SISTEMA MÍNIMO E GRAVADOR (para as cpu's AT89S51, AT89S52, AT89S53 e AT89S8252)



## 4.5 AGV – VEÍCULO GUIADO AUTOMATICAMENTE (detalhamento do projeto)

### 4.5.1 Fluxograma de Controle



## 4.5.2 LISTAGEM DO SOFTWARE

```
2500 A.D. 8051 Macro Assembler  -  Version 4.00j
        ------------------------------------------------

            Input  Filename : agv20.asm
            Output Filename : agv20.obj

 1                      ;#########################################################################
 2                      ;Projeto:           AGV - VEÍCULO GUIADO AUTOMATICAMENTE    versão: 2.0
 3                      ;Autor:             Wilson Ruiz                              19/11/03
 4                      ;Descrição:         Protótipo de um veículo transportador (triciclo)
 5                      ;                   microcontrolado, com sensores de posicionamento, um motor
 6                      ;                   DC responsável pelo deslocamento longitudinal e um motor
 7                      ;                   de passo para a correção da trajetória definida por uma
 8                      ;                   linha clara pintada sobre um piso escuro.
 9                      ;ALGORÍTMO DE CONTROLE:
10                      ;A.         RESET
11                      ;  A1.      Desliga motores
12                      ;  A2.      Apaga LEDs de status
13                      ;  A3.      Configurações internas do microcontrolador (desabilita interrupções)
14                      ;  A4.      Inicializa as variáveis do sistema (byte espelho de M0, byte
15                      ;           status dos sensores, contador de passos e satck pointer)
16                      ;B.         POSICIONAMENTO AXIAL/FRONTAL
17                      ;           B1.     Espera acionamento da tecla INÍCIO
18                      ;           B2.     Leitura dos sensores
19                      ;           B3.     Atualiza LEDs status e byte de status
20                      ;           B4.     Verifica sensores
21                      ;                   B4.1    Se SDE ou SDD = V, vai para B2
22                      ;                   B4.2    Se SPE ou SPD ou S1 = F, vai para B2
23                      ;C.         INICIALIZAÇÃO DE M0
24                      ;           C1.     Espera acionamento da tecla INÍCIO
25                      ;           C2.     Carrega contador de passos (R3) com valor de 40 passos
26                      ;           C3.     Carrega contador de passos (R4) com valor de 80 passos
27                      ;           C4.     Verifica S0
28                      ;                   C4.1    Se S0 = F vai para C4.13
29                      ;                   C4.2    Move M0 um passo no sentido horário e atualiza byte espelho
30                      ;                   C4.3    Verifica S0
31                      ;                   C4.4    Se S0 = F vai para C4.13
32                      ;                   C4.5    Decrementa contador R3
33                      ;                   C4.6    Se contador > 0 volta para C4.2
34                      ;                   C4.7    Verifica S0
35                      ;                   C4.8    Se S0 = F vai para C4.13
36                      ;                   C4.9    Move M0 um passo no sentido anti-horário e atualiza byte espelho
37                      ;                   C4.10   Decrementa contador R4
38                      ;                   C4.11   Se contador > 0 volta para C4.9
39                      ;                   C4.12   Aciona rotina de falha de S0 (LED piscando)
40                      ;                   C4.13   Atualiza LEDs dos sensores e byte status
41                      ;                   C4.14   Zera o contador de passos
42                      ;D.         OPERAÇÃO
43                      ;           D1.     Espera acionamento da tecla INÍCIO
44                      ;           D2.     Aciona M1
45                      ;           D3.     Leitura dos sensores
46                      ;           D4.     Atualiza LEDs dos sensores e byte status
47                      ;           D5.     Verifica sensores axiais
48                      ;                   D5.1    Se SDE = V vai para D5.3
49                      ;                   D5.2 CORREÇÃO DIANTEIRA PARA A DIREITA
50                      ;                   D5.3 Se SDD = V vai para D6.1
51                      ;                   D5.4 CORREÇÃO DIANTEIRA PARA A ESQUERDA
52                      ;           D6.     Verifica sensores frontais
53                      ;                   D6.1    Se SPE = F vai para D6.3
54                      ;                   D6.2    Parada de 30 segundos
55                      ;                   D6.3    Se SPD = F vai para D6.5
56                      ;                   D6.4    Parada de 1 minuto
57                      ;                   D6.5    Se S1 = F vai para D6.8
58                      ;                   D6.6    Desliga M1
59                      ;                   D6.7    Leitura da tecla INVÍCIO
60                      ;                           D6.7.1  Se tecla não acionada volta para D6.6
61                      ;                           D6.7.2  Aciona M1
62                      ;                   D6.8    Volta para D5
63                      ;E.         CORREÇÃO DIANTEIRA PARA A DIREITA
64                      ;           E1.     Atualiza status correspondente
```

```
65                  ;           E2.     Lê byte espelho de M0
66                  ;           E3.     Atualiza byte espelho de M0
67                  ;           E4.     Gira M0, no sentido horário, um passo
68                  ;           E5.     Verifica sensor SDE
69                  ;                   E5.1    Se SDE = F vai para E2
70                  ;           E6.     Lê byte espelho de M0
71                  ;           E7.     Atualiza byte espelho de M0
72                  ;           E8.     Gira M0, no sentido anti-horário, um passo
73                  ;           E9.     Verifica S0
74                  ;                   E.9.1   Se S0 = F vai para E6
75                  ;           E10.    Atualiza Leds dos sensores e byte status
76                  ;           E11.    Retorna
77                  ;F.         CORREÇÃO DIANTEIRA PARA A ESQUERDA
78                  ;           F1.     Atualiza status correspondente
79                  ;           F2.     Lê byte espelho de M0
80                  ;           F3.     Atualiza byte espelho de M0
81                  ;           F4.     Gira M0, no sentido anti-horário, um passo
82                  ;           F5.     Verifica sensor SDD
83                  ;                   F5.1    Se SDD = F vai para F2
84                  ;           F6.     Lê byte espelho de M0
85                  ;           F7.     Atualiza byte espelho de M0
86                  ;           F8.     Gira M0, no sentido horário, um passo
87                  ;           F9.     Verifica S0
88                  ;                   F.9.1   Se S0 = F vai para F6
89                  ;           F10.    Atualiza Leds dos sensores e byte status
90                  ;           F11.    Retorna
91                  ;Principais variáveis:      R0:  byte espelho de M0
92                  ;                           R1:  byte rascunho de M0
93                  ;                           R2:  byte dos LEDs de status dos sensores
94                  ;                           R3:  contador de passos
95                  ;                           R4:  contador de passos
96                  ;                           R5:  uso para temporização
97                  ;                           R6:  uso para temporização
98                  ;                           R7:  uso para temporização
99                  ;                           A:   uso geral
100                 ;                           B:   uso geral
101                 ;Descrição do hardware:     Microcontrolador 8751, operando em 12Mhz.
102                 ;                           Sensores ópticos ligados a comparadores.
103                 ;                           de tensão.
104                 ;                           Circuito de acionamento de motor DC.
105                 ;                           Circuito de acionamento de motor de passo.
106                 ;                           LED's indicadores de status.
107                 ;                           Dip switch e teclas.
108                 ;           Portas:
109                 ;           P0: Entrada dos sensores
110                 ;               p/ sensor = V = 0 para faixa ou marca detectada
111                 ;                   P0.7 ← S0: Alinhamento do motor de passo
112                 ;                   P0.6 ← S1: Parada definitiva
113                 ;                   P0.5 ← SPE: Parada esquerda (temporizada)
114                 ;                   P0.4 ← SPD: Parada direita (temporizada)
115                 ;                   P0.3 ← SDE: Dianteiro esquerdo
116                 ;                   P0.2 ← SDD: Dianteiro direito
117                 ;           P1: Saída para controle dos motores
118                 ;                   P1.7 → M1: acionado para bit = 1
119                 ;                   P1.6
120                 ;                   P1.5
121                 ;                   P1.4
122                 ;                   P1.3 → fase A de M0          A  B   A' B'
123                 ;                   P1.2 → fase B de M0          0  0   0  1 passo 1
124                 ;                   P1.1 → fase A' de M0         0  0   1  0 passo 2
125                 ;                   P1.0 → fase B' de M0         0  1   0  1 passo 3
126                 ;                                                1  0   0  0 passo 4
127                 ;           P2: Saída para os LEDs indicadores de status dos sensores
128                 ;             LED aceso para bit = 1
129                 ;               P2.7
130                 ;               P2.6
131                 ;               P2.5 → LED do sensor SDD
132                 ;               P2.4 → LED do sensor S0
133                 ;               P2.3 → LED do sensor SDE
134                 ;               P2.2 → LED do sensor SPE
135                 ;               P2.1 → LED do sensor S1
136                 ;               P2.0 → LED do sensor SPD
137                 ;           P3: Entrada para a dip switch e tecla INÍCIO
138                 ;               P3.7 ← Tecla INÍCIO, quando acionada = 1
139                 ;               P3.6 ← dip switch número 4
```

```
140                         ;                   P3.5 ← dip switch número 3
141                         ;                   P3.4
142                         ;                   P3.3
143                         ;                   P3.2
144                         ;                   P3.1 ← dip switch número 2
145                         ;                   P3.0 ← dip switch número 1
146                         ;Arquivo fonte:     AGV20.asm
147                         ;Programas:         X8051:              Compilador assembly
148                         ;                   Link:               Ligador
149                         ;                   AVSIM51:            Simulador
150                         ;#########################################################################
151
152                         ;************** A = INÍCIO ***************************************************
153  0000                                       ORG    0000H                ;reset
154  0000  02 00 50                            LJMP     INICIO
155
156  0050                                       ORG    0050H                ;início do programa
157  0050  75 90 00         INICIO:            MOV      P1, #00000000B     ;desliga M0 e M1
158  0053  75 A0 00                             MOV      P2, #00000000B     ;apaga LEDs de status
159  0056  75 A8 00                             MOV      IE, #00000000B     ;desabilita todas interrupções
160  0059  75 81 30                             MOV      SP, #0030H
161  005C  78 11                                MOV      R0, #11H           ;byte espelho de M0
162  005E  79 00                                MOV      R1, #00H           ;rascunho do byte espelho de M0
163  0060  7A 00                                MOV      R2, #00H           ;byte status dos sensores
164  0062  7B 00                                MOV      R3, #00H           ;contador de passos
165  0064  7C 00                                MOV      R4, #00H           ;contador de passos
166
167                         ;************** B = POSICIONAMENTO AXIAL/FRONTAL ***************
168  0066  20 B7 FD         L1:                JB       P3.7, L1           ;espera tecla INÍCIO
169  0069  E5 80            L2:                MOV      A, P0              ;lê sensores
170  006B  FA                                  MOV      R2, A              ;atualiza byte status sensores
171  006C  31 79                               ACALL    ATUALIZA           ;atualiza LEDs de status
172  006E  30 A1 F8                            JNB      P2.1, L2           ;S1    (verifica os sensores
173  0071  30 A2 F5                            JNB      P2.2, L2           ;SPE    e se o AGV está  fora
174  0074  30 A0 F2                            JNB      P2.0, L2           ;SPD    da posição axial /
175  0077  20 A3 EF                            JB       P2.3, L2           ;SDE    frontal inicial
176  007A  20 A5 EC                            JB       P2.5, L2           ;SDD    volta)
177
178                         ;************** C = INICIALIZAÇÃO DE M0 ******************************
179
180  007D  20 B7 FD         L20:               JB       P3.7, L20          ;espera tecla INÍCIO
181  0080  7B 28                               MOV      R3, #28H           ;contador com 40 = 28H passos
182  0082  7C 50                               MOV      R4, #50H           ;contador com 80 = 50H passos
183  0084  20 87 13                            JB       P0.7, L21          ;verifica S0
184  0087  31 0E            L22:               ACALL    MOVE_M0_H          ;move M0 no sentido horário
185  0089  20 87 0E                            JB       P0.7, L21          ;verifica S0
186  008C  DB F9                               DJNZ     R3, L22            ;decrementa e se > 0 volta
187  008E  20 87 09                            JB       P0.7, L21          ;verifica S0
188  0091  31 1F            L23:               ACALL    MOVE_M0_AH         ;move M0 no sentido anti-horário
189  0093  20 87 04                            JB       P0.7, L21          ;verifica S0
190  0096  DC F9                               DJNZ     R4, L23            ;decrementa e se > 0 volta
191  0098
192  0098  31 E2                               ACALL    FALHA_S0
193
194  009A  E5 80            L21:               MOV      A, P0              ;lê sensores
195  009C  FA                                  MOV      R2, A              ;atualiza byte status sensores
196  009D  31 79                               ACALL    ATUALIZA           ;atualiza LEDs de status
197  009F
198  009F  7B 00                               MOV      R3, #00H           ;zera contador de passos
199  00A1  7C 00                               MOV      R4, #00H
200
201                         ;************** D = OPERACAO *************************************
202  00A3  20 B7 FD         L24:               JB     P3.7, L24            ;espera tecla INÍCIO
203  00A6  D2 97                               SETB     P1.7               ;aciona M1
204  00A8  E5 80            L32:               MOV      A, P0              ;leitura dos sensores
205  00AA  FA                                  MOV      R2, A              ;atualiza byte status
206  00AB  31 79                               ACALL    ATUALIZA           ;atualiza LEDs de status
207  00AD
208  00AD  30 A3 02                            JNB      P2.3, L25          ;verifica SDE
209  00B0  11 DC                               ACALL  VIRA_FRENTE_D        ;correção dianteira direita
210
211  00B2  30 A5 02         L25:               JNB      P2.5, L26          ;verifica SDD
212  00B5  11 F5                               ACALL  VIRA_FRENTE_E      ;correção dianteira esquerda
213  00B7
214  00B7  20 A2 08         L26:               JB       P2.2, L27          ;verifica SPE
```

```
215  00BA   C2 97                          CLR     P1.7              ;desliga M1
216  00BC   31 5F                          ACALL   DELAY_10          ;parada de 10 segundos
217  00BE   D2 97                          SETB    P1.7              ;aciona M1
218  00C0   31 C8                          ACALL   DELAY_1           ;pausa de 1 segundo
219  00C2
220  00C2   20 A0 0A        L27:           JB      P2.0, L29         ;verifica SPD
221  00C5   C2 97                          CLR     P1.7              ;desliga M1
222  00C7   31 5F                          ACALL   DELAY_10          ;pausa de 20 segundos
223  00C9   31 5F                          ACALL   DELAY_10
224  00CB   D2 97                          SETB    P1.7              ;aciona M1
225  00CD   31 C8                          ACALL   DELAY_1           ;pausa de 1 segundo
226  00CF
227  00CF   20 A1 07        L29:           JB      P2.1, L31         ;verifica S1
228  00D2   C2 97                          CLR     P1.7              ;desliga M1
229  00D4   20 B7 CC                       JB      P3.7, L24         ;espera tecla INÍCIO
230  00D7   D2 97                          SETB    P1.7              ;aciona M1
231  00D9
232  00D9   02 00 A8        L31:           LJMP    L32               ;volta
233  00DC
234                         ;#########################################################################
235                         ;subrotina:                VIRA_FRENTE_D
236                         ;descrição:                Correção dianteira para a direita
237                         ;                          Atualiza status correspondente
238                         ;                          Chama MOVE_M0_H
239                         ;                          Verifica sensor SDE e se este = F
240                         ;                          continua correção
241                         ;                          Chama MOVE_M0_AH
242                         ;                          Verifica S0 e se este = F continua
243                         ;                          correção contrária
244                         ;                          Atualiza Leds sensores e byte status
245                         ;entradas:
246                         ;saídas:
247                         ;variáveis internas:
248                         ;hardware:
249                         ;OBS:
250                         ;#########################################################################
251  00DC   C0 E0           VIRA_FRENTE_D: PUSH    A
252  00DE   E5 80                          MOV     A, P0             ;leitura dos sensores
253  00E0   31 79                          ACALL   ATUALIZA          ;atualiza LEDs de status
254  00E2   FA                             MOV     R2, A             ;atualiza byte status
255  00E3   31 0E           L35:           ACALL   MOVE_M0_H
256  00E5   20 83 FB                       JB      P0.3, L35
257  00E8   31 1F           L36:           ACALL   MOVE_M0_AH
258  00EA   30 87 FB                       JNB     P0.7, L36
259  00ED   E5 80                          MOV     A, P0             ;leitura dos sensores
260  00EF   31 79                          ACALL   ATUALIZA          ;atualiza LEDs de status
261  00F1   FA                             MOV     R2, A             ;atualiza byte status
262  00F2   D0 E0                          POP     A
263  00F4   22                             RET
264
265                         ;#########################################################################
266                         ;subrotina:                VIRA_FRENTE_E
267                         ;descrição:                Correção dianteira para a esquerda
268                         ;                          Atualiza status correspondente
269                         ;                          Chama MOVE_M0_AH
270                         ;                          Verifica sensor SDD e se este = F
271                         ;                          continua correção
272                         ;                          Chama MOVE_M0_H
273                         ;                          Verifica S0 e se este = F continua
274                         ;                          correção contrária
275                         ;                          Atualiza Leds sensores e byte status
276                         ;entradas:
277                         ;saídas:
278                         ;variáveis internas:
279                         ;hardware:
280                         ;OBS:
281                         ;#########################################################################
282  00F5   C0 E0           VIRA_FRENTE_E: PUSH    A
283  00F7   E5 80                          MOV     A, P0             ;leitura dos sensores
284  00F9   31 79                          ACALL   ATUALIZA          ;atualiza LEDs de status
285  00FB   FA                             MOV     R2, A             ;atualiza byte status
286  00FC   31 1F           L38:           ACALL   MOVE_M0_AH
287  00FE   20 82 FB                       JB      P0.2, L38
288  0101   31 0E           L39:           ACALL   MOVE_M0_H
289  0103   30 87 FB                       JNB     P0.7, L39
```

```
290  0106  E5 80                    MOV     A, P0               ;leitura dos sensores
291  0108  31 79                    ACALL   ATUALIZA            ;atualiza LEDs de status
292  010A  FA                       MOV     R2, A               ;atualiza byte status
293  010B  D0 E0                    POP     A
294  010D  22                       RET
295
296                  ;#########################################################################
297                  ;subrotina:                MOVE_M0_H
298                  ;descrição:                Lê byte espelho de M0
299                  ;                          Gira M0, no sentido horário, um passo
300                  ;                          Atualiza byte espelho
301                  ;entradas:
302                  ;saídas:
303                  ;variáveis internas:
304                  ;hardware:
305                  ;OBS:
306                  ;#########################################################################
307  010E  E8        MOVE_M0_H:     MOV     A, R0               ;lê byte espelho
308  010F  03                       RR      A                   ;desloca byte espelho
309  0110  F8                       MOV     R0, A               ;atualiza byte espelho
310  0111  54 0F                    ANL     A, #00001111B
311  0113  F9                       MOV     R1, A
312  0114  E5 90                    MOV     A, P1
313  0116  54 F0                    ANL     A, #11110000B
314  0118  49                       ORL     A, R1
315  0119  F5 90                    MOV     P1, A               ;move M0 um passo
316  011B  31 30                    ACALL   DELAY               ;pausa
317  011D  49                       ORL     A, R1
318  011E  22                       RET
319  011F
320                  ;#########################################################################
321                  ;subrotina:                MOVE_M0_AH
322                  ;descrição:                Lê byte espelho de M0
323                  ;                          Gira M0, no sentido anti-horário, um passo
324                  ;                          Atualiza byte espelho
325                  ;entradas:
326                  ;saídas:
327                  ;variáveis internas:
328                  ;hardware:
329                  ;OBS:
330                  ;#########################################################################
331  011F  E8        MOVE_M0_AH:    MOV     A, R0               ;lê byte espelho
332  0120  23                       RL      A                   ;desloca byte espelho
333  0121  F8                       MOV     R0, A               ;atualiza byte espelho
334  0122  54 0F                    ANL     A, #00001111B
335  0124  F9                       MOV     R1, A
336  0125  E5 90                    MOV     A, P1
337  0127  54 F0                    ANL     A, #11110000B
338  0129  49                       ORL     A, R1
339  012A  F5 90                    MOV     P1, A               ;move M0 um passo
340  012C  31 30                    ACALL   DELAY               ;pausa
341  012E  49                       ORL     A, R1
342  012F  22                       RET
343
344                  ;#########################################################################
345                  ;subrotina:                DELAY
346                  ;descrição:                Fornece uma pausa entre os passos de M0
347                  ;entradas:
348                  ;saídas:
349                  ;variáveis internas:
350                  ;hardware:
351                  ;OBS:                      Necessária para o ajuste da velocidade do motor de passo
352                  ;#########################################################################
353  0130  C0 05     DELAY:         PUSH    R5
354  0132  C0 06                    PUSH    R6
355  0134  C0 07                    PUSH    R7
356  0136  C0 E0                    PUSH    A
357  0138  E5 B0                    MOV     A, P3               ;lê a posição da dip switch
358  013A  54 03                    ANL     A, #00000011B       ;verifica posições de 1 e 2
359  013C  FD                       MOV     R5, A               ;carrega no contador R5
360  013D  E5 B0                    MOV     A, P3               ;lê a posição da dip switch
361  013F  54 60                    ANL     A, #01100000B       ;verifica posições de 3 e 4
362  0141  03                       RR      A                   ;desloca byte da dip switch
363  0142  03                       RR      A
364  0143  03                       RR      A
```

```
365  0144  4D                          ORL   A, R5              ;compara com R5
366  0145  FD                          MOV   R5, A              ;carrega no contador R5
367  0146  7E 25          L42:        MOV   R6, #25H           ;carrega contador R6
368  0148  7F 25          L41:        MOV   R7, #25H           ;carrega contador R7
369  014A  1F              L40:        DEC   R7                 ;decrementa R7
370  014B  BF 00 FC                    CJNE  R7, #00H, L40
371  014E  1E                          DEC   R6                 ;decrementa R6
372  014F  BE 00 F6                    CJNE  R6, #00H, L41
373  0152  1D                          DEC   R5                 ;decrementa R5
374  0153  BD 00 F0                    CJNE  R5, #00H, L42
375  0156  D0 07                       POP   R7
376  0158  D0 05                       POP   R5
377  015A  D0 06                       POP   R6
378  015C  D0 E0                       POP   A
379  015E  22                          RET
380
381                        ;###########################################################################
382                        ;subrotina:              DELAY_10
383                        ;descrição:              Fornece uma pausa de aproximadamente
384                        ;                        10 segundos (com clock do sistema de 12 MHz)
385                        ;entradas:
386                        ;saídas:
387                        ;variáveis internas:
388                        ;hardware:
389                        ;OBS:                    equação:
390                        ;                        84 + 204.X + 132.X.Y + 60.X.Y.Z = 120000000
391                        ;                        onde X, Y e Z são os valores em
392                        ;                        R5, R6 e R7 respectivamente
393                        ;###########################################################################
394  015F  C0 E0           DELAY_10:   PUSH  A
395  0161  7D FF                       MOV   R5, #FFH           ;carrega valor X
396  0163  7E 57           L47:        MOV   R6, #57H           ;carrega valor Y
397  0165  7F 57           L46:        MOV   R7, #57H           ;carrega valor Z
398  0167  EF              L45:        MOV   A, R7
399  0168  14                          DEC   A
400  0169  FF                          MOV   R7, A
401  016A  70 FB                       JNZ   L45
402  016C  EE                          MOV   A, R6
403  016D  14                          DEC   A
404  016E  FE                          MOV   R6, A
405  016F  70 F4                       JNZ   L46
406  0171  ED                          MOV   A, R5
407  0172  14                          DEC   A
408  0173  FD                          MOV   R5, A
409  0174  70 ED                       JNZ   L47
410  0176  D0 E0                       POP   A
411  0178  22                          RET
412
413                        ;###########################################################################
414                        ;subrotina:              ATUALIZA
415                        ;descrição:              Atualiza LEDs de status dos sensores
416                        ;entradas:
417                        ;saídas:
418                        ;variáveis internas:
419                        ;hardware:
420                        ;OBS:
421                        ;###########################################################################
422  0179  20 87 05        ATUALIZA:   JB    P0.7, L3
423  017C  C2 A4                       CLR   P2.4
424  017E  02 01 83                    LJMP  L4
425  0181  D2 A4           L3:         SETB  P2.4
426  0183  20 86 05        L4:         JB    P0.6, L5
427  0186  C2 A1                       CLR   P2.1
428  0188  02 01 8D                    LJMP  L6
429  018B  D2 A1           L5:         SETB  P2.1
430  018D  20 85 05        L6:         JB    P0.5, L7
431  0190  C2 A2                       CLR   P2.2
432  0192  02 01 97                    LJMP  L8
433  0195  D2 A2           L7:         SETB  P2.2
434  0197  20 84 05        L8:         JB    P0.4, L9
435  019A  C2 A0                       CLR   P2.0
436  019C  02 01 A1                    LJMP  L10
437  019F  D2 A0           L9:         SETB  P2.0
438  01A1  20 83 05        L10:        JB    P0.3, L11
439  01A4  C2 A3                       CLR   P2.3
```

```
440  01A6  02 01 AB                    LJMP    L12
441  01A9  D2 A3          L11:         SETB    P2.3
442  01AB  20 82 05       L12:         JB      P0.2, L13
443  01AE  C2 A5                       CLR     P2.5
444  01B0  02 01 B5                    LJMP    L14
445  01B3  D2 A5          L13:         SETB    P2.5
446  01B5  20 81 05       L14:         JB      P0.1, L15
447  01B8  C2 A6                       CLR     P2.6
448  01BA  02 01 BF                    LJMP    L16
449  01BD  D2 A6          L15:         SETB    P2.6
450  01BF  20 80 03       L16:         JB      P0.0, L17
451  01C2  C2 A7                       CLR     P2.7
452  01C4  22                          RET
453  01C5  D2 A7          L17:         SETB    P2.7
454  01C7  22                          RET
455
456                       ;#########################################################################
457                       ;subrotina:              DELAY_1
458                       ;descrição:              Fornece uma pausa de aproximadamente
459                       ;                        1 segundo (com clock do sistema de 12 MHz)
460                       ;entradas:
461                       ;saídas:
462                       ;variáveis internas:
463                       ;hardware:
464                       ;OBS:                    equação:
465                       ;                        84 + 204.X + 132.X.Y + 60.X.Y.Z = 12000000
466                       ;                        onde X, Y e Z são os valores em
467                       ;                        R5, R6 e R7 respectivamente
468                       ;#########################################################################
469  01C8  C0 E0          DELAY_1:     PUSH    A
470  01CA  7D FF                       MOV     R5, #FFH        ;carrega valor X
471  01CC  7E 1B          L53:         MOV     R6, #1BH        ;carrega valor Y
472  01CE  7F 1B          L52:         MOV     R7, #1BH        ;carrega valor Z
473  01D0  EF             L51:         MOV     A, R7
474  01D1  14                          DEC     A
475  01D2  FF                          MOV     R7, A
476  01D3  70 FB                       JNZ     L51
477  01D5  EE                          MOV     A, R6
478  01D6  14                          DEC     A
479  01D7  FE                          MOV     R6, A
480  01D8  70 F4                       JNZ     L52
481  01DA  ED                          MOV     A, R5
482  01DB  14                          DEC     A
483  01DC  FD                          MOV     R5, A
484  01DD  70 ED                       JNZ     L53
485  01DF  D0 E0                       POP     A
486  01E1  22                          RET
487
488                       ;#########################################################################
489                       ;subrotina:              FALHA_S0
490                       ;descrição:              Indica falha no posicionamento de M0
491                       ;entradas:
492                       ;saídas:                 LED de status do S0 piscando
493                       ;variáveis internas:
494                       ;hardware:
495                       ;OBS:
496                       ;#########################################################################
497  01E2  00             FALHA_S0:    NOP
498  01E3  D2 A7          L55:         SETB    P2.7
499  01E5  31 C8                       ACALL   DELAY_1
500  01E7  C2 A7                       CLR     P2.7
501  01E9  31 C8                       ACALL   DELAY_1
502  01EB  21 E3                       JMP     L55
503  01ED
504  01ED                              END

      Lines Assembled :  504          Assembly Errors :  0
```

### 4.5.3 O formato HEXA intel

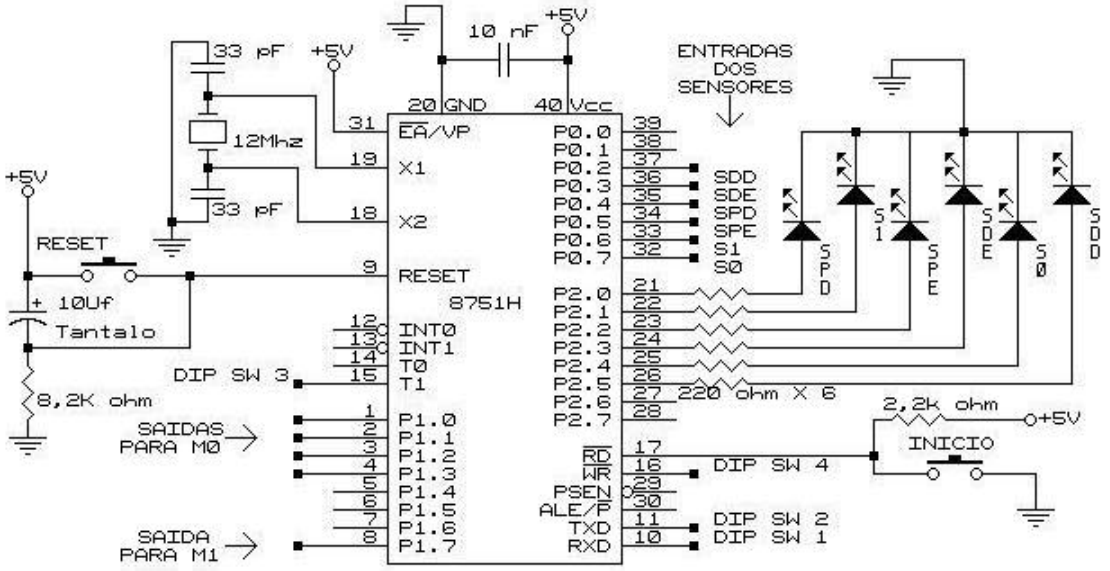
Todos os arquivos fornecidos para a gravação seguem o formato hexa intel abaixo.
Cada linha do arquivo mostrado indica:

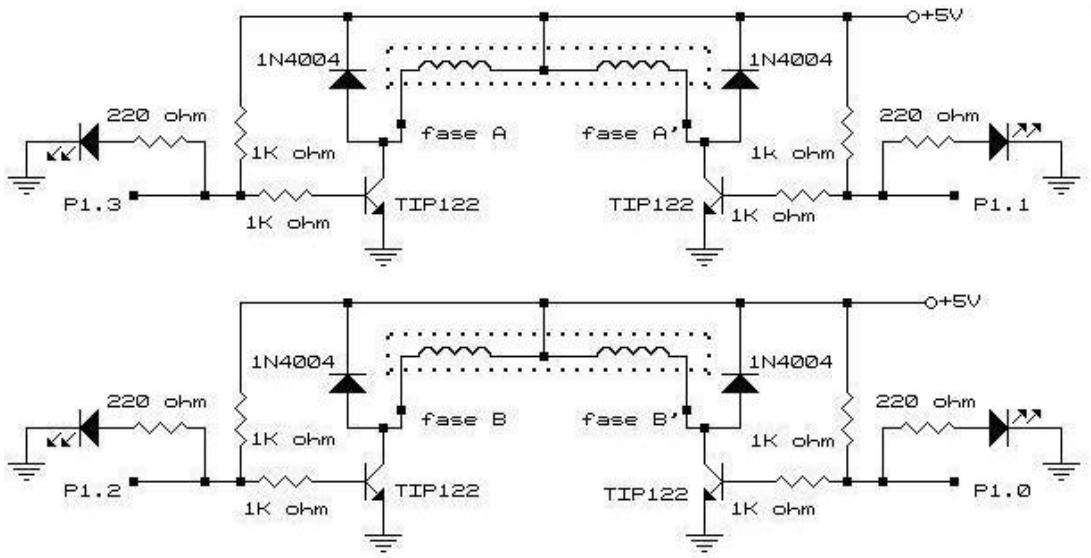
- O início sempre é com :
- No primeiro campo é indicado a quantidade de bytes (em hexadecimal) presentes na linha.
- No segundo campo é indicado o endereço do primeiro byte da linha.
- O terceiro campo é sempre 00.
- No quarto campo temos os bytes do programa.
- No quinto campo, um byte de checksum (CS), para verificação de integridade do arquivo.
- A última linha serve apenas como referência de final de arquivo.
- O arquivo abaixo é mostrado em colunas apenas para facilitar a identificação dos respectivos campos.

O arquivo AGV20.HEX

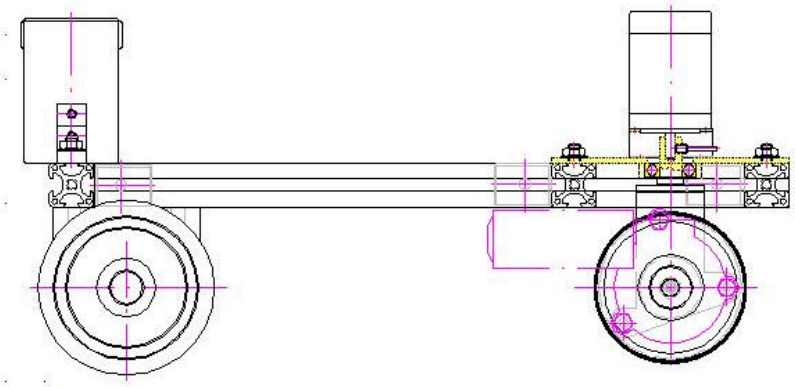
```
:03   0000  00    020050                              AB
:10   0050  00    75900075A00075A80075813078117900    41
:10   0060  00    7A007B007C0020B7FDE580FA317930A1    71
:10   0070  00    F830A2F530A0F220A3EF20A5EC20B7FD    C8
:10   0080  00    7B287C50208713310E20870EDBF92087    D8
:10   0090  00    09311F208704DCF931E2E580FA31797B    F0
:10   00A0  00    007C0020B7FDD297E580FA317930A302    B9
:10   00B0  00    11DC30A50211F520A208C297315FD297    5A
:10   00C0  00    31C820A00AC297315F315FD29731C820    72
:10   00D0  00    A107C29720B7CCD2970200A8C0E0E580    64
:10   00E0  00    3179FA310E2083FB311F3087FBE58031    F7
:10   00F0  00    79FAD0E022C0E0E5803179FA311F2082    20
:10   0100  00    FB310E3087FBE5803179FAD0E022E803    3D
:10   0110  00    F8540FF9E59054F049F59031304922E8    50
:10   0120  00    23F8540FF9E59054F049F59031304922    05
:10   0130  00    C005C006C007C0E0E5B05403FDE5B054    FB
:10   0140  00    600303034DFD7E257F251FBF00FC1EBE    FF
:10   0150  00    00F61DBD00F0D007D005D006D0E022C0    CB
:10   0160  00    E07DFF7E577F57EF14FF70FBEE14FE70    AB
:10   0170  00    F4ED14FD70EDD0E022208705C2A40201    49
:10   0180  00    83D2A4208605C2A102018DD2A1208505    BB
:10   0190  00    C2A2020197D2A2208405C2A00201A1D2    6C
:10   01A0  00    A0208305C2A30201ABD2A3208205C2A5    71
:10   01B0  00    0201B5D2A5208105C2A60201BFD2A620    A8
:10   01C0  00    8003C2A722D2A722C0E07DFF7E1B7F1B    37
:10   01D0  00    EF14FF70FBEE14FE70F4ED14FD70EDD0    23
:0D   01E0  00    E02200D2A731C8C2A731C821E338

:00   0000  01FF
```

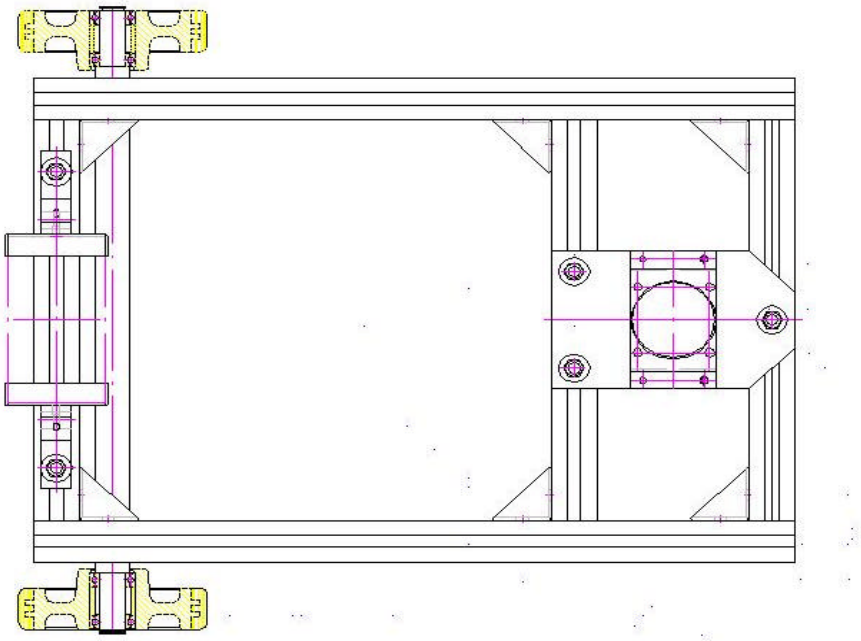
### 4.5.4 ESQUEMAS ELETRÔNICOS



### 4.5.5 DESENHOS MECÂNICOS





**BIBLIOGRAFIA:**


- Microcontrôleurs 8051 et 8052
  Description et mise em oeuvre
  Bernard Odant
  Dunod Tech – Paris, France

- Aplicações Práticas do Microcontralador 8051
  Teoria geral detalhada
  Vidal Pereira da Silva Jr.
  Editora Érica

- Microcontroladores 8051
  Salvador P. Gimenez
  Prentice Hall

- Manuais Intel

- Manuais Philips

- Manuais Atmel

- Notas de aulas diversas

- Foto da capa: Wilson Ruiz