UNIVERSIDADE EDUARDO MONDLANE

FACULDADE DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELECTROTÉCNICA

Microprocessadores

Engº. Albino B Cuinhane

AULA 5
SUMÁRIO

Capítulo 5

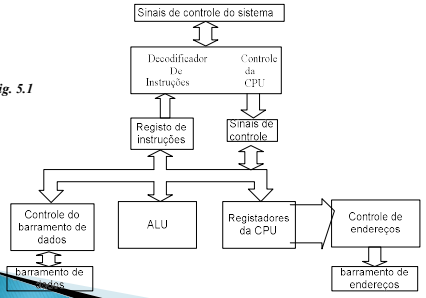
Programação Em Assembly – caso do Z80

5.1. Modelo de Programação

5.1.1. Elementos Principais

- Para programarmos o Z80, ou outro microprocessador, devemos ter em mente a sua organização interna. Voltemos a lembrar o diagrama de funcional.

Fig. 5.1

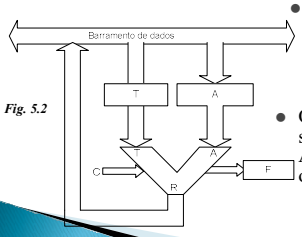


5.1.1. Elementos Principais

- Antes de avançarmos vamos olhar de perto a ALU, o Acumulador e o registo de Bandeiras por serem elementos com funções especiais e que os mencionaremos várias vezes.

ALU-Aritmetic Logic Unit.

Fig. 5.2



- A ALU trabalha em estreita colaboração dos registos T e A.
- O T representa o registo temporário (B, C, D, E, H ou L) onde pode ser guardado momentaneamente o segundo operando
- O registo A é o Acumulador, onde se conserva um dos operandos(o A) e o resultado da operação entre os operandos A e T

5.1.1. Elementos Principais

ALU-Aritmetic Logic Unit.

- A ALU recebe sinais de controle, C, da unidade de controle
- A ALU sinaliza o estado das operações que realiza, activando uma série de bandeiras que compõem o registo F
- Os dados vêm e vão sempre no barramento de dados. Os sinais de controle é que definem com exactidão o uso do barramento

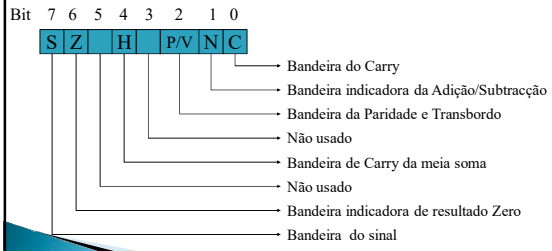
Acumulador

- O Acumulador é um registo de propósito geral adistrito à ALU. Nele é colocado um dos operandos e depois volta-se a colocar nele o resultado da operação fornecido pela ALU

5.1.1. Elementos Principais

Registo de Bandeiras

- O registo de bandeira serve para sinalizar certas condições relevantes para o funcionamento do sistema, e a prossecução do programa. É composto por 8 bits como se segue. A descrição das bandeiras é feita no ponto 5.3.



5.1.1. Elementos Principais

Bandeira do sinal, S

A bandeira do sinal é activada pela ULA para indicar o sinal do número resultante da operação aritmética de números com sinal

A bandeira S conserva o estado do bMS do acumulador (bit 7). Nas operações aritméticas é usado o complemento 2 e o bit 7 é igual a 0 para numero positivo e é igual a 1 para negativo.

Resulta disto que o diapasão da magnitude representável extende-se de -128 a +127, usando os 6 bits restantes.

Quando se introduz um byte do dispositivo de E/S a bandeira S indica igualmente o sinal do byte.

5.1.1. Elementos Principais

Bandeira do Zero, Z

Esta bandeira é normamlemente activada quando o resultado duma operação resulta em 0:

a) Para operações aritmeticas e lógicas de 8 bits, a Z é igual a 1 se o byte resultante no acumulador é igual 0.

b) Para instruções de comparação (Search) a Z = 1 se o valor no acumulador é igual ao do local da memoria indicado pelo par de registos HL.

c) Quando se testa um bit num registo ou num local de memoria a bandeira Z contém o complemento do bit testado

d) Quando se trocam dados (INI, IND, OUTI e OUTD) entre um periférico e a memória a bandeira Z é igual a 1 se o conteudo do registo B for nulo.

5.1.1. Elementos Principais

Bandeira da Meia-Soma, H

Esta bandeira é colocada em 1 ou 0 dependendo se "Vai 1" ou "Vem 1" que sucedem entre os bits 3 e 4 nas operações aritmeticas.

É usada também nas instrução DAA para corrigir o resultado em BCD na soma e subtração.
H é usada como se segue:

H	Adição	Subtração
1	Ocorreu um "vai 1" do bit 3 para o bit 4	Ocorreu um "vem 1" do bit 4
0	Não correu um "vai 1" do bit 3 para o bit 4	Não ocorreu um "vem 1" do bit 4

5.1.1. Elementos Principais

Bandeira da Paridade/Transbordo, P/V

Esta bandeira é usada de acordo com a operação que está sendo executada. Para operações aritmeticas ela indica o transbordo quando o resultado da operação sai do diapasão possivel no acumulador (-128 a +127). Nessa condição é colocada em 1

Esta bandeira é usada também nas operações lógicas e rotações para indicar que o resultado obtido tem a paridade Par. È então colocada em 1 e na paridade Impar é zerada

Esta bandeira é usada noutras aplicações durante as instruções CPI, CPIR, CPD, CPDR, LDI, LDIR, LDD, LDDR para indicar o estado do contador de bytes. Quando este contador for nulo a bandeira P/V é também zerada

Quando introduzido um byte dum periférico, a bandeira é usada para indicar a paridade do dado

5.1.1. Elementos Principais

Bandeira da Adição/Subtracção, N

A bandeira Adição/Subtracção N é usada na instrução Decimal Adjust Accumulator (DAA) para distinguir as instruções ADD e SUB. Para instrucção ADD, a bandeira N é zerada e na SUB é posta em 1.

5.1.1. Elementos Principais

Bandeira do Vai/Vem Um, C

Esta bandeira é colocada em 1 nos casos em que a adição de dois numeros gera um "Vai 1" ou quando a operação da subtração gera um "Vem 1". De contrário é zerado

A instrução DAA, que ajusta o Acumulador para soma e subtração em BCD, também coloca a bandeira C em 1 quando a condição de ajuste decimal for encontrada.

Nas insturções RLA, RRA, RLS e RRS, a banerira Carry é usada como elo entre Bms e o BMS dum registo ou memória uma vez que nestes deslocamentos a saída do acumulador é feita através desta bandeira.

Nas instruções logicas AND, OR e XOR, a bandeira é zerada

5.1.1. Elementos Principais

CONCLUSÃO SOBRE AS BANDEIRAS

Uma vez que as bandeiras sinalizam condições especiais que ocorrem ao realizar as operações aritméticas e lógicas, o programador faz uso delas para controlar o fluxo dum programa.

Por exemplo, ao realizar a multiplicação de dois numeros fazem-se somas repetidas do multiplicando (M) *multiplicador* vezes menos 1 (m-1). Enquanto não se fazer m-1 somas o processo se repete

A forma de controlar m-1 vezes é decrementar m (m-1). no momento em a operação m-1=0 a bandeira de Zero é activada e o programador usa esse aviso para terminar o ciclo DO.

5.1.2. Estrutura e Organização dum programa em assembly

Endereço	Conteúdo	Mnemonico	Operando	Comentário
1)	2)	3)	4)	5)
1)	2)	3)	4)	5)
1) Local da memória onde o conjunto de bits que constituem a instrução serão colocados				
2) Conjunto de bits que estão no local de memória com o endereço dado em 1)				
3) Letras que mimam a palavra inglesa que diz a operação a ser feita				
4) Elementos a serem manipulados pela instrução				
5) Comentários para facilitar a documentação				
NOTA: PARA SIMPLIFICAR, O 3) E 4) FUNDEM-SE NA MESMA COLUNA				

5.1.2. Estrutura e Organização dum programa em assembly

EXEMPLO 1:

Endereço	Conteúdo	Mnemonico	Operandos	Comentário
5000	3E	LD	A,01	Para colocar 01 no acumulador
5001	01			Como o mnemónico ocupou 1 byte o 01 está a seguir na memória
5002	06	LD	B,02	Para colocar 02 no registo B
5003	02			

EXEMPLO 2:

Endereço	Conteúdo	Instrução	Comentário
5000	3E	LD A,01	Para colocar 01 no acumulador
5001	01		Como o mnemónico ocupou 1 byte o 01 está a seguir na memória
5002	06	LDB,02	Para colocar 02 no registo B
5003	02		

5.2. Tipos de Instruções

5.2. Tipo de Instruções

- O microprocessador executa operações com base nas instruções que busca na memória, sequencialmente.
- Uma instrução é uma sequência de bits(tensão 0 ou 1) que são descodificadas no Decodificador de Instruções que converte em condição de transição para a Unidade de Controle.
- A unidade de controle por sua vez activa as linhas de tarefas respectivas de acordo com a condição de transição. Então todas as unidades realizam tarefas específicas o que resulta numa operação adequada dos bits.
- As instruções podem ser classificadas de diversas maneiras, mas podemos distinguir 5 categorias:
  - Transferência de dados,
  - Processamento de dados,
  - Testes e saltos,
  - Entrada/Saida e
  - Controle

5.2.1. Instruções de Transferência De Dados

As instruções de transferência de dados movimentam dados entre registos ou entre registos e memória.

O formato destas instruções normalmente têm indicação da direção do movimento, em que se indica a origem dos dados e o destino.

EXEMPLO:
LD A, (001A) que move dados do local 001AH para o acumulador

A movimentação de dados não altera os que estão na origem.

È possivel movimentar dados num bloco de memória com uma única instrução. Esta operação é executada por regresso sequenciado à memória e buscar os dados um a um.

5.2.2. Instruções de Processamento de dados

As instruções de Processamento de dados podem ser subdivididas ainda em 5 categorias gerais:

- Operações aritméticas
- Manipulação de bits
- Incremento e decremento
- Operações lógicas
- Deslocamento e rotação

As operações aritméticas operam (soma e subtração) dado existente no acumulador(um dos operandos) com outro num dpr registos temporários ou memória e devolve o resultado para o acumulador.
Na operação são activadas diversas bandeiras conforme o caso

5.2.2. Instruções de Processamento de dados

A manipulação de bits inclui activar um bit em 1 ou zerá-lo, testar um bit, fazer deslocamentos para esquerda ou direita e rotações

O incremento e decremento envole o acréscimo ou redução numa unidade ao conteúdo dum registo ou memória.

As operações lógicas são as já conhecidas funções AND, OR , NOT e XOR

Deslocações e Rotações inclui operações de deslocamentos para esquerda ou direita e rotações

EXEMPLOS:
1. ADD A,0A que adiciona o dado 0AH com o conteúdo do acumulador
2. SET 2,B que coloca em 1 o bit 2 do registo B
3. SLA C que desloca o conteúdo do registo C para a esquerda. O Bit 7 vai para a bandeira do Carry e o bit 0 é zerado.

5.2.3. Instruções De Testes e Saltos

As instruções de teste verificam o valor dum bit especificado no registo e levam a UCP tomar uma decisão conforme o caso.

As instruções de salto estão estrictamente ligadas as de teste de bits. Verificam uma condição e levam a UCP para um determinado ponto do programa.

Os saltos podem ser divididos em dois tipos: O salto e o desvio.

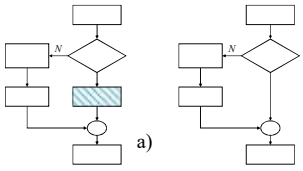
•Num salto a execução do progama interrompe a sequencia linear se a condição for satisfeita e passa para outro ponto. Não executa mas a instrução seguinte à do salto.

•Num desvio a execução do programa sofre uma interrupção momentânea mas será continuada pela instrução seguinte quando terminar a execução da causa do desvio.

5.2.3. Instruções de Testes e Saltos

Desvios E Saltos

A Fig. 5.3 mostra uma situação de salto a) e de desvio b).



a) b).

Fig. 5.3

- *EXEMPLOS:*
- *1. JP NZ,LOOP o conteúdo do PC salta para o local LOOP se a bandeira do Zero é Zero*
- *2. JP M,(01BA) o conteúdo do PC salta para 01BAH se a bandeira do Sinal é Zero*
- *3. CALL nn o conteudo do PC salta para a localização da primeira instrução da rotina*

5.2.4. Instruções de Entrada e saída de dados

- Os periféricos estão ligados aos barramentos como os dispositivos internos. De modo que a CPU os vê como memória. As instruções de I/O diferem das internas pelo facto de que devem ter em conta a velocidade de processamento dos periféricos
- Nestas instruções o Bms do endereço indica a porta pela qual será feita a comunicação

5.2.5. Instruções de Control

- As instruções de controle permitem a selecção de diferentes condições e tipos de processamento da CPU

5.3. Acervo de Instruções do Z80

5.3. Acervo de Instruções do Z80

- A intenção deste parágrafo é introduzir o conjunto de instruções do Z80 e a forma como são constituídas
- A instrução é composta por duas partes:

(1) o Mnemónico ou OpCode que é uma abreviatura especial para a operação a ser executada e

(2) o Operando (que pode ser em forma de endereço do local onde vai ser encontrado o Operando):

- LD A,01 – Carrega o acumulador com o número 01

- LD A,(5000H) – Carrega o acumulador com o dado existente no local 500H

- LD(5000H),A – Carrega a memória no local 500H com o conteúdo do acumulador

5.3. Acervo de Instruções do Z80

- Entretanto existem instruções com uma parte apenas. São os seguintes:
  - HALT
  - NOP
  - RET
- A indicação do local da memória pode ser feito de várias maneiras:
  - **ENDEREÇAMENTO DIRECTO**
  - **ENDEREÇAMENTO INDIRECTO**
  - **ENDEREÇAMENTO INDEXADO**
  - **ETC.**

5.3. Acervo de Instruções do Z80

MNEMONICO	FUNÇÃO
ADC	Adiciona com vai 1 (carry)
ADD	Adiciona
AND	Produto lógico
BIT	Testa o bit
CALL	Chama sub-rotina
CCF	Bandeira do carry suplementar
CP	Compara
CPD	Compara o Acumulador com memória, decresce endereço e o contador de byte (CNTB)
CPDR	Compara o Acumulador com memória, decresce endereço e o CNTB, continua até que a condição seja encontrada ou CNTB=0
CPI	Compara o Acumulador com memória, decresce CNTB e incrementa o endereço
CPIR	Compara o Acumulador com memória, decresce CNTB e incrementa o endereço, continua até que a condição seja encontrada ou CNTB=0

MNEMONICO	FUNÇÃO
CPL	Complementa o Acumulador
DAA	Ajuste decimal
DEC	Decresce
DI	Inabilita as interrupções
DJNZ	Decresce o registo B, salta se não for zero
EI	Habilita as interrupções
EX	Troca
HALT	Suspende o programa
IM	Define o modo de interrupção
INC	Incrementa
IND	Carregar na memória e decresce o ponteiro
INDR	Carregar na memória e decrescer o ponteiro até que CNTB=0
INI	Carregar na memória e incrementa o ponteiro
INIR	Carregar na memória e incrementa o ponteiro até que CNTB=0
JP	Salto
JR	Salto Relativo

5.3. Acervo de Instruções do Z80

MNEMONICO	FUNÇÃO
LD	Carregar
LDD	Transfere dados entre locais de Memória, decresce os endereços do destino e da origem
LDDR	Transfere dados entre locais de Memória até que CNTB=0, decresce os endereços do destino e da origem
NEG	Negar o conteúdo do Acumulador
NOP	Não fazer nada
OR	Soma lógico
OTDR	Tira os dados na memoria para o exterior, decresce o endereço, continue até que o B=0
OTIR	Tira os dados na memoria para o exterior, incrementa o endereço, continue até que o B=0
OUT	Enviar dados para uma porta
OUTD	Enviar dados para uma porta, decrescer o endereço
OUTI	Enviar dados para uma porta,incrementar o endereço

MNEMONICO	FUNÇÃO
POP	Buscar dados na pilha
PUSH	Guardar dados na pilha
RES	Zerar um bit
RET	Regressar da sub-rotina
RETI	Regressar do pedido de interrupção
RETN	Regressar duma interrupção não mascarada
RL	Rodar para esquerda
RR	Rodar para direita
RST	Reiniciar
SBC	Subtrair com carry
SCF	Pôr em 1 a bandeira do Carry
SET	Pôr em 1 um bit
SLA	Deslocamento para esquerda aritmético
SRA	Deslocamento para direita aritmético
SRL	Deslocamento para direita lógico
SUB	Subtrair
XOR	OU exclusivo

5.3 Acervo de Instruções do Z80

- **Exemplo 1**. Carregar o acumulador com o número 01, o registo B com 02 e o registo C com 03

Endereço	Conteúdo	Mnemonico	Operando	Comentário
5000	3E	LD	A,01	Para colocar 01 no acumulador
5001	01			Como o mnemónico ocupou 1 byte o 01 está a seguir na memória
5002	06	LD	B,02	Para colocar 02 no registo B
5003	02			Como o mnemónico ocupou 1 byte o 02 está a seguir na memória
5004	0E	LD	C,03	Para colocar 03 no registo C
5005	03			Como o mnemónico ocupou 1 byte o 03 está a seguir na memória

- A coluna do conteúdo mostra a informação real (em Hex) que o compilador vai guardar na memória. Portanto é um conjunto de bits sem sentido para o humano.

5.3 Acervo de Instruções do Z80

- A coluna do conteúdo representa o formato da instrução armazenada na memória. É chamada Opcode
- Porém, na verdade os dados estão armazenados em binário.
- A maioria das instruções ocupam mais que um local de memória uma vez que o Mnemónico ocupa 1 byte e o operando 1 ou 2 bytes

Endereço	Conteúdo	Mnemonico	Operando	Binário
5000	3E	LD	A,01	0011 1110
5001	01			0000 0001

- Felizmente os compiladores tratam destes aspectos da utilização eficaz da memória, sem contudo dizer que o programador não deve ter cuidado, pois pode receber muitas mensagens de erro na altura da compilação

5.3 Acervo de Instruções do Z80

- **Exemplo 2**. Adicionar 22 e 11 e guardar o resultado no local 5012. Os números estão armazenados nos locais 500A e 500F respectivamente.
- Para resolver este problema temos que perceber como o computador realiza as operações. Lembramos que ele faz passo a passo e temos que ter sempre em mente as instruções possíveis que nos ajudam a realizar o nosso algoritmo
- O algoritmo (um dos possíveis) será:
  - 1. buscar o numero que está no endereço 500A e guardar no acumulador.
  - 2. Carregar o par de registo HL com o endereço do segundo numero. Este esta guardado no local 500F
  - 3. somar o conteúdo do acumulador com o dado residente no local 500F. O resultado fica no acumulador
  - 4. guardar o conteúdo do acumulador no local 5012

5.3 Acervo de Instruções do Z80

Duração duma instrução e bandeiras afectadas

Na litertura relativa ao Z80 é indicado o conjunto de recursos(físicos e tempo) usados em cada instrução.

No livro "Introduction to Microcomputers, MICAMASTER 980 & 960", que será a nossa referência, as instruções estão assim organizadas:

Mnemónico da instrução Descrição do que faz a instrução

Code Indicação do código da instrução em Hexadecimal

Flag Lista de todas as 8 bandeiras

Status Indicação das bandeiras afectadas pela instrução. Se Y a bandeira é afectada
Nota: Adicionamente é indicada a quantidade de ciclos T consumidos pela instrução.

5.3 Acervo de Instruções do Z80

A duração duma instrução é indicada em termos de ciclos T. O valor final deste tempo depende da frequência de Ck.

As bandeiras afectadas permitem-nos fazer o teste delas e tomar decisões

Exemplo:

SUB	data						
Code	D6 data						
Flag	S	Z	H	P/O	N	C	Clock cycles
Status	Y	Y	Y	Y	1	Y	= 7

Esta instrução tem Opcode = D6 seguido do dado de 1 Byte, todas as bandeiras são afectadas sendo que a N é sempre igual a 1. A duração total é de 7 ciclos T

5.4. Decisões

5.4. Decisões

A resolução de problemas com o computador envolve uma séries de decisões que se tomam ao longo do programa.

Para tomar a decisão passa-se necessariamente por um teste duma certa condição.

Ex. Quando escurecer acender a luz,
Se a temperatura for baixa ligar o aquecedor

Na verdade toda a acção do processador é uma decisão que depende duma condição.

Infelismente, o processador não tem olhos para ver que está escuro nem pele para sentir a temperatura, etc.

Para que o processador tome uma decisão, tem que ser simulada a situação e representada por uma palavra binária. É esta palavra que vai ser testada.

5.4. Decisões

As decisões são tomadas com base no teste de bits. Se o bit testado for 0 vai-se para um lado, de contrário vai-se para outro lado.

As bandeiras são os bit primários que são testados para criar um salto condicional

Uma das instruções usadas para verificar uma condição dada é o CP n que compara o conteudo do Acumulador com um valor n.

Se o dado n for igual ao conteúdo do Acumulador a bandeira Z será colocada em 1. Este é o bit que desve ser testado para orientar a decisão.

Exemplo:
Para testar que está escuro, colocamos um sensor de luz que gera o bit 1 quando está escuro. Se estiver ligado ao bit 1 na porta paralela, podemos testá-lo com CP 01(ou seja CP 0000 0001 em binário). Se tivermos já a entrada disponível para comparação, o bit Z será 1 se estiver escuro e 0 no caso contrário. A condição Z=1 pode ser usada para um salto condicional a um local do programa que activa um bit de saída que actua no interruptor da lâmpada

5.4. Decisões

As instruções de salto condicional testam condições específicas das bandeiras. Se a condição testada NÃO EXISTE o programa continua como se nada tivesse acontecido. Se a condição EXISTE o programa salta para um local de endereço especificado

INSTRUÇÃO	OPCODE	FUNÇÃO
JP NZ, ENDR	C2	Salto para o ENDeReço se a bandeira Z = 0
JP Z, ENDR	CA	Salto para o ENDeReço se a bandeira Z = 1
JP NC, ENDR	D2	Salto para o ENDeReço se a bandeira C = 0
JP C, ENDR	DA	Salto para o ENDeReço se a bandeira C = 1
JP P, ENDR	E2	Salto para o ENDeReço se a bandeira S = 0
JP M, ENDR	EA	Salto para o ENDeReço se a bandeira S = 1
JP PO, ENDR	F2	Salto para o ENDeReço se a bandeira P/O = 0
JP PE, ENDR	FA	Salto para o ENDeReço se a bandeira P/O = 1
JR NZ, DESL	20	Salto relativo por um DESLocamento se Z=0
JR Z, DESL	28	Salto relativo por um DESLocamento se Z=1
JR NC, DESL	30	Salto relativo por um DESLocamento se C=0
JR C, DESL	38	Salto relativo por um DESLocamento se C=1

5.5. PIO - Parallel Input/Output Device

5.5.1. Arquitectura do PIO

A entrada e saída de dados para o ambiente do Z80 é feita através de portas paralelas ou seriais

A memória ROM e "RAM" (no nível Cache e Principal) são os utentes principais dos barramentos de dados e de endereço. Funcionam à alta velocidade e facilmente dialogam com a CPU.

Por este motivo a entrada e saída de dados para estes dispositivos é feita directamente. Aliás, deve ser feita desta maneira pois esta parte da memória é usada para conservar os dados de uso temporário

Quando se trata de receber ou entregar dados do exterior a questão muda. Porque é lidar com dispositivos periféricos é necessário encontrar uma forma de adaptar tanto as velocidades como a forma de transporte (se paralelo ou serial) de dados.

5.5.1. Arquitectura do PIO

Um dos dispositivos usados para isso é o PIO que é programável pela UCP para ser interfacear um grande número de periféricoas com impressoras, teclado, mostradores, etc.

A entrada/saída de dados é feita através do circuito PIO – Perallel Input Output que comporta duas portas de 8 bits cada. Como a PIO funciona no modo paralelo, todos os 8 bits são recebido/entregues ao mesmo tempo

Uma das caracteristicas de destaque do PIO é o circuito de controle de interrupções que permite uma interação eficaz entre a UCP e os periféricos sem necessidade de circuitos externos adicionais.

Além disso, depedendo da preprogramação, o PIO pode interromper a UCP se condições especiais pré-fixadas ocorrem no dispositivo periférico (por exemplo uma tecla premida, pode levar a UCP a atender o teclado)

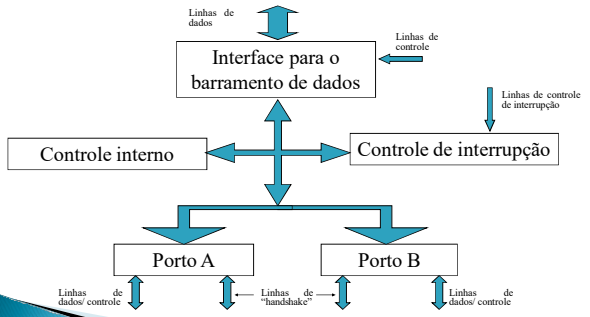
5.5.1. Arquitectura do PIO

Características da PIO

- Todas as entradas e saídas são compatíveis com TTL
- Alimentado a 5V e uma única fase de Ck
- Possui dois portos completamente independentes com o linhas bidirecionais cada um, para inteface com periféricos
- Possui um controle de transferência de dados com linhas "handshake" controlando o fluxo de dados em ambos os sentidos "simultaneamente"
- Possui uma interrupção em série ("Daisy Chain") em que se programa um vector que coloca os periféricos em série conforme a prioridade
- Pode ser programa para operar em 4 modos:
  - MODO 0 – Byte de saída
  - MODO 1 – Byte de entrada
  - MODO 2 – Byte bidirecional
  - MODO 3 – Bit controlado

5.5.1. Arquitectura do PIO

Arquitectura Geral Da PIO



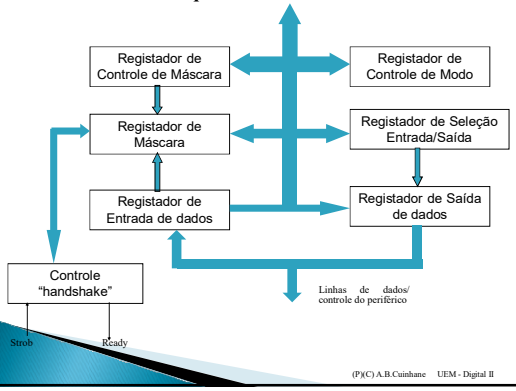
5.5.1. Arquitectura do PIO

Arquitectura Geral Do PIO

O PIO consiste internamente dum circuito de interface com o baramento de dados, um de controle interno, um circuito de controle de interrupções e os circuitos dos portos em si.
O PIO pode interfacear a UCP directamente com qualquer periférico mas se a quantidade destes for elevado poderá ser necessário circuitos externos adicionais como descodificadores de endereço ou registos temporários("buffer").

5.5.1. Arquitectura do PIO

Arquitectura Interna Do Porto



5.5.1. Arquitectura do PIO

Arquitectura Interna Do Porto

Os dois portos são completamente idênticos e são compostos por 6 registos gerais e um circuito "hand-shake":
• 1 registo de 8 bits para entrada de dados
• 1 registo de 8 bits para saída de dados
• 1 registo de 8 bits para o vector da mâscara
• 1 registo de 8 bits para gestão do sentido(saida/entrada)
• 1 registo de 2 bits para controle do modo e
• 1 registo de 2 bits para controle de máscara

5.5.1. Arquitectura do PIO

Arquitectura Interna Do Porto

Registos de Entrada e Saída de Dados:
A transferencia de dados entre o a UCP e o periférico é feita através destes registos. Quando a UCP quer entregar dados ao periférico, escreve-os no registo de saída. Quando o periférico quer entregar dados à UCP escreve-os no registo de entrada.

Registo Direccional (gestão do sentido):
É usado no modo 3 para definir o sentido em cada Bit. Cada bit deste registo corresponde uma linha do barramento de dados.
Se o bit deste registo for 0 a linha correspondente é saída, se for 1 a linha será entrada.

Controle "handshake"
Determina a estrutura de interupções no sistema UCP-Periférico através da geração da cadeia de prioriade de interruptores. O dispositivos prioritários podem atrair para si a atenção dos menos.

5.5.1. Arquitectura do PIO

Arquitectura Interna Do Porto

Registo Máscara/Controle da Máscara:
O registo Máscara também é usado no modo 3 juntamente com a interrupção que tem sua ocorrência relacionada com condições especiais do periférico.
Neste registo é pragramado a palvra (ou vector) que corresponde à condição que se deseja observar. Se um bit, ou conjunto de bit, satisfazer uma condição será gerada uma interrupção.
A geração da interrupção pode ser no modo AND, quando todos os bits forem activos ou modo OR, quando pelo menos um estiver activo

Registo de Controle de Modo:
É programada pela UCP de modo a seleccionar um dos 4 modos de funcionamento da PIO

5.5.2 Programação das Entradas e Saídas

O sistema utilizado no laboratório de Digital, o MIC960/MAT980 usa o dispositivo Z8536 que comporta além do PIO o CTC

A entrada e saída de dados é feita através das portas A e B que são vistas como locais de memória pelo processador. Portanto têm endereços

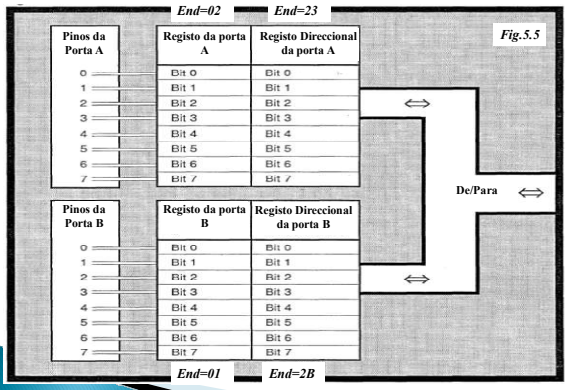
Os pinos das portas são bidirecionais. Mas apontam para uma direção de cada vez e é possível programá-los individualmente como foi visto na descrição do PIO. Normalemente são todas entradas.

Destaquemos em cada porta dois registos: o Registo de dados e o Direcional (Fig.5.5).

O registo Direccional determina quais pinos são entrada e quais são saída

O registo de dados guarda dados que entram/saem da porta. É um registo bidireccional

5.5.2 Programação das Entradas e Saídas



5.5.2 Programação das Entradas e Saídas

- A programação é feita através do Registo de Controle cujo endereço é 03

- Dois Passos para configurar uma porta

1. Escrever o endereço do Registo Direcional (23 ou 2B) no Registo de Controle (03). Isto faz com que Registo de Controle aponte para o Registo Direcional desejado (23 ou 2B).
2. O valor que se deseja escrever no Registo Direcional (23 ou 2B) é escrito no Registo de Controle (03) o que na verdade coloca os dados no Registo Direcional (23 ou 2B) e programa os pinos.

- A programação dos pinos bidirecionais funciona de acordo com o seguinte critério:
  - Saída: Bit correspondente no Registo de Controle = 0
  - Entrada: Bit correspondente no Registo de Controle = 1

5.5.3 Programação das Entradas e Saídas

- *Exemplo: Programar o porto A como saída nos 4 bMs e entradas nos 4 bms (ou seja 0000 1111)*

Instrução	Comentário
LD A,23	Carrega o Acumulador com um dado (23) que corresponde ao endereço do registo direcional do porto A (RDA)
OUT (03),A	Envia para o Registo de Controle (03) o número 23. isto aponta para o RDA
LD A,0F	Carrega o Acumulador com um dado (0F)
OUT (03),A	Envia para o Registo de Controle (03) o numero que está no acumulador (0F) . Isto programa os pinos do porta A

5.6. Subrotinas

5.6.Subrotinas

Sub-rotina e funções são programas residentes fora do programa principal, PP, e que são invocados quando são necessárias.

A sub-rotina é especializada numa tarefa específica. Tem dados de entrada e de saída.

Os dados de entrada e de saída da sub-rotina estão localizadas em locais específicos da memória: tem nomes. O programa principal deve ter "conhecimento" desses locais e prepara-los antes de activar a sub-rotina.

Quando a sub-rotina termina, os dados de saída estão prontos e poderão ser buscados pelo PP no local previamente preparado.

Nas linguagens de alto nível, a invocação da sub-rotina é feita pelo seu nome. Já em assembly, a invocação da sub-rotina, tal como do PP, é feita através do endereço da residência da primeira instrução.

5.6.Subrotinas

Exemplos:

Exemplo 1. fazer um programa que multiplica 2 números P1 e P2 e gera o produto P.

Resolução:

a) Reservar 3 lugares: multiplicando, multiplicador e o produto
b) Dar nomes a esses lugares
c) Elaborar o algoritmo/fluxograma do produto.

Adoptaremos:
Local de P1: 4200;
Local de P2: 4201;
Local de P: 4202
Local inicial da Sub-rotina da Multiplicação:4210

5.6.Subrotinas

Exemplos:

Exemplo 2. fazer um programa que divide 2 números D1 e D2 e gera o Quociente Q.

Resolução:

a) Reservar 3 lugares: Dividendo, Divisor e o Quociente
b) Dar nomes a esses lugares
c) Elaborar o algoritmo/fluxograma do produto

Adoptaremos:
Local de D1: 4300;
Local de D2: 4301;
Local de D: 4302
Local inicial da Sub-rotina da Divisão:4310

5.6.Subrotinas

Exemplos:

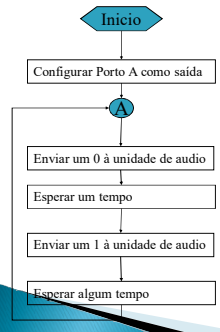
Exemplo 3. fazer um programa que calcula $y(x)=2x^2+3x+1$ para um valor de x guardado no local 4001. o valor de y deve ser guardado em 4002.

Os estudantes devem realizar usando o conceito de subrotina

5.7. Exemplos de Aplicação

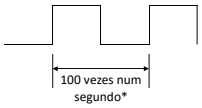
5.7.1. Controle duma Unidade de Audio

- *Escrever um programa que gere um sinal de 1KHz através dum altofalante*



Tudo o que precisamos é mover a membrana do altofalante para frente e para trás 1000 vezes em 1 segundo. O fluxograma a seguir nos mostra o caminho a seguir

O tempo a esperar deve ser tal que num período de 1 segundo seja enviado 1000 "Zeros" e 1000 "Uns"



* portanto, duração de 0,500ms em High e 0,500ms em Low

5.7.1. Controle duma Unidade de Audio

Para resolvermos isso devemos primeiro saber qual programa vamos escrever.

Verificamos quanto tempo consome esse programa.

Introduzimos rotinas de perda de tempo que nos permitam encaixar mais tempo para preenchermos todo o intervalo que desejamos.

O cálculo do tempo é com base nos ciclos T que as instruções consomem multiplicado pelo tempo que dura cada ciclo.

No MIC960 o relógio é de 4MHz portanto $T=(4x10^6Hz)^{-1} = 0,25\mu s$

O programa que desejamos deve ir à saída em cada 500µs mudar o valor (a partir do ponto A do fluxograma).

Portanto deve ir em cada 2000 (=500/0,25) ciclos T.

Para que a unidade de áudio sinta os sinais, ligámo-la à um dos pinos do porto A, digamos bit0.

Logo, enviar 00, zera o bit0 e enviao 01 faz set do bit0.

Etiqu.	Instru	Comentário
	LD A,23	Configurado porto A como saída em todos os bits
	OUT (03),A	
	LD A,00	
	OUT(03),A	
Inicio	LD A,00	7 ciclos T
Sair 0	OUT(02),A	11xT
	LD A,01	7xT
Sair 1	OUT(02),A	11XT
	JP Inicio	10XT

O programa leva 18 T de Sair 0 a Sair 1 e leva 28 T de Sair 1 a Sair 0. Ou seja faltam 1982 T no primeiro intervalo e 1972 T no segundo intervalo. Aqui entra o conceito de subrotinas. Dai que fazemos a modificação ao lado para sair do programa principal e rodar um subrotina que consome tempo.

Nota: O tempo que a subrotina consome deve ter em conta que a instrução RET também consome tempo.

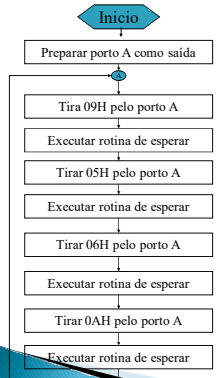
Etiq.	Instrução	Comentário
	LD A,23	
	OUT (03),A	
	LD A,00	
	OUT(03),A	
Inicio	LD A,00	
Sair 0	OUT(02),A	
	LD A, Tempo 1	a)
	CALL Atraso	b)
	LD A,01	
Sair 1	OUT(02),A	
	LD A,Tempo 2	a)
	CALL Atraso	
	JP Inicio	

a) Carrega o acumulador com um valor a ser usado pela subrotina para perda de tempo

b) Chama a subrotina que usa o dado no acumulador para perder tempo

TPC: escrever a subrotina para perda de tempo

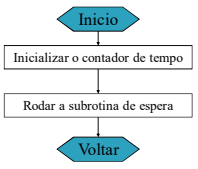
5.7.2. Controle dum Motor de Passos



Escrever um programa que governe um motor de passos

Para resolver o problema seguiremos os seguintes passos:

1. Perceber como funciona o motor de passos
2. Relizar o programa que executa o fluxograma ao lado



Rotina de esperar

5.7.2. Controle dum Motor de Passos

O que importa reter é que o motor possui uma entrada de 4 bits.

A combinação de 1 e 0 nos 4 bits magnetiza partes do estator e força o rotor a mover num certo ângulo.

Para que se mova aos saltos(Motor de passos!) deve mover-se e parar

Esta é a razão porque o programa puxa pela subrotina de espera.

Pormenores deste programa podem ser vistos no livro:

*Bradley, John - **Introduction to Microprocessors Micamaster 980 & 960,** Feedback, England*

.