UNIVERSIDADE EDUARDO MONDLANE

FACULDADE DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELECTROTÉCNICA

Microprocessadores

Engº. Albino B Cuinhane

AULA 3
SUMÁRIO

3. Modos de Endereçamento, Ciclos de Tempo e Interrupções
- 3.1. Modos de endereçamento
- 3.2. Ciclos de Tempo
- 3.3. Interrupções

3.1. Modos de Endereçamento

3.1. Modos de Endereçamento

- Muitas instruções envolvem operações de números que estão nos registos, memória ou Dispositivos de E/S. Modo de endereçamento refere-se á forma como a origem e destino destes dados é feita na instrução.
- A indicação do local ou dado a aceder pode ser feita de várias maneiras mas destacaremos:
- ENDEREÇAMENTO IMEDIATO quando na instrução já aparece o dado com que se deve trabalhar.
  Ex. LD A, 42 que carrega o acumulador com o dado 42.
  Vendo esta instrução não se questiona mais qual será o conteúdo do acumulador assim que for executada

3.1. Modos de Endereçamento

- ENDEREÇAMENTO DIRECTO quando na instrução aparece já o local da memória.
  Ex. LD A,(4200) que carrega o acumulador com o dado existente no local da memória 4200 (de referir que é 4200H).
  Vendo esta instrução não sabemos qual será o conteudo de A mas sabemos logo onde está o valor
- ENDEREÇAMENTO INDIRECTO quando o local da memória é indicado indirectamente através do conteúdo dum par(porquê?) de registo.
  Ex. LD A,(HL) que carrega o acumulador com o dado existente no local da memória formado pelo conteúdo dos registos HL. H contém o byte mais significativo e L o menos significativo
  Vendo esta instrução não sabemos qual será o conteudo de A nem onde está o valor. Temos que primeiro pedir a H e L para nos indicarem o local e lá espreitarmos o valor.

3.1. Modos de Endereçamento

- ENDEREÇAMENTO INDEXADO quando o local da memória é indicado indirectamente através dum numero que deve ser adicionado ao valor de um dos Registos de Indice IX ou IY. Os valores destes registos não é alterado após a execução da instrução. No entanto o PC recebe um novo valor obtido nesta operação e com isso aponta para novo local de memória
  Ex. BIT 2, (IX+ ***d***) que testa o bit 2 do local da memória indicado pelo valor IX somado ao ***d*** sinalizado em complemento a 2.
- ENDEREÇAMENTO IMPLÍCITO quando operação ditada pela instrução traz implícito um ou mais registos internos da CPU.
  Ex. ADD B.
  Nesta instrução está implícito que o segundo operando está no acumulador

3.1. Modos de Endereçamento

- ENDEREÇAMENTO RELATIVO quando o local da memória é indicado indirectamente através dum numero que deve ser adicionado ao Program Counter. Este modo sucede nos casos de saltos

  Ex. JR 0A que carrega o Program Counter com um valor dado pelo valor actual mais 0A.

- ENDEREÇAMENTO POR REGISTO quando a instrução contem registos internos da CPU como sendo os locais a serem usados na operação

  Ex. LD A, B que carrega o dado contido no registo B para o A.

- Outros tipos de endereçamento pode ser usados. São o caso do Indexado, Pagina Zero, implicito, Indirecto por registo. Recomendo a ler o manual fornecido nas aulas:

Zilog – Z80 Family CPU User Manual - UM008005-0205, 2004, San Jose

3.2. Ciclos de Tempo

3.2. Ciclos de Tempo

O Z80, assim como outro processador, realiza as instruções através duma sequência precisa de passos básicos, que são:

- Leitura e escrita da memória
- Leitura e escrita de periféricos
- Reconhecimento de interrupções

Qualquer e todas as instruções consome tempo. A unidade básica deste tempo é o ciclo de relógio chamado Ciclo T (Vide Figura a seguir). Tal como referimos antes, em cada ciclo T algo acontece.

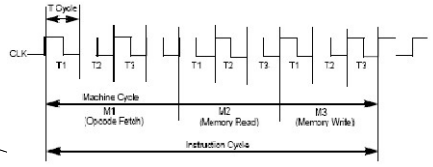
Os ciclos de operação do processador são chamados Ciclo Máquinas M1, M2 e M3.

No primeiro ciclo, M1, é feita a leitura da memória apenas para buscar o Mnemónico da instrução. Este ciclo é chamado "Fetch" ou Busca

3.2. Ciclos de Tempo

O ciclo M1 pode durar até 6 ciclos T, a menos que seja alongado por um sinal WAIT. Este sinal é usado pelos elementos mais lentos que a UCP para avisá-lo de que ainda não estão prontos para receber ou entregar dados.

Os ciclos M2 e M3 são usados para movimentar dados entre á UCP e a memória ou periférico e cada um pode durar entre 3 a 6 ciclos T. Também podem ser alongados se houver um pedido de espera.



Ciclos T e M

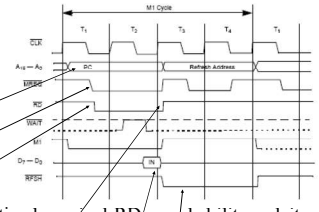
3.2. Ciclos de Tempo

No flanco descendente do ciclo T2 e subsequentes, dentro de cada ciclo M, a UCP verifica se a linha WAIT está activa. Se tal for é adicionado um estado de espera no próximo ciclo T para permitir que o solicitante da espera esteja pronto

Durante o ciclo de espera, a UCP pode fazer outras tarefas que não afectem o elemento que solicitou a espera.

3.2. Ciclos de Tempo

Ciclo de Busca



1. O ciclo de busca começa com a colocação do conteúdo do CP(Contador de Programa) no barramento de endereços.

2. No flanco descendente seguinte a UCP activa a linha MREQ

3. No mesmo momento é activado o sinal RD para habilitar a leitura da memória.

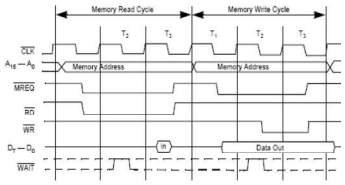
4. Os dados ficam diponíveis no barramento de dados e no próximo flanco ascendente de Ck a UCP reconhece-os. Logo a seguir as linhas MREQ e RD são desactivadas

5. Os ciclos T3 e T4 são para refrescar as memórias dinâmicas, descodificar e executar a instrução que se buscou

3.2. Ciclos de Tempo

Ciclo de Escrita/Leitura da memória

Os ciclos de escrita e leitura são similares ao fetch. O sinal MREQ e RD são usados do mesmo modo



No ciclo de escrita o sinal MREQ é activado depois que os dados no barramento de endereços estiverem estabilizados

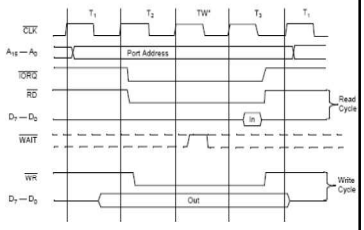
No ciclo de leitura o sinal RD é activado depois que os dados no barramento de dados estejam estabilizados.

Este sinal deve ser desactivado antes de mudar os dados no barramento de endereços para evitar sobreposição de dados a serem lidos no barramento de dados

3.2. Ciclos de Tempo

Ciclo de Entrada/Saida

Durante o ciclo de Entrada ou Saída de dados para periférico um sinal WAIT é sempre acrescentado. Isto é necessário pois o tempo que a UCP precisa para ler/escrever dados é muito mais curto que a reação dos periféricos



Repare-se que na verdade a escrita/leitura de periférico não é feito directamente neste, mas sim nas portas de E/S.

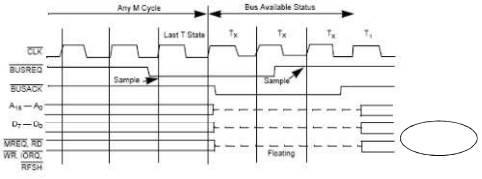
3.2. Ciclos de Tempo

Ciclo de Pedido/Reconhecimento de barramento

A arquitectura dos computadores permite a partilha de recuros ligados no mesmo barramento, para flexibilizar processos, libertar a UCP e aproveitar o tempo em que esta não precisa dos barramentos.

Por isso é previsto uma linha BUSREQ que é activado pelos dispositivos externos à UCP.

No flanco ascendente de cada ciclo máquina a UCP verifica se a linha BUSRQ está activa. Se tal for, coloca as suas saídas para os barramentos em alta impedância e deixa que o elemento solicitade do barramento faca uso deste.



3.3. Interrupções

- O atendimento de periféricos pode ser feito de dois mecanismos:
  - ***Polling e Interrupção***
- O polling consiste na observação sistemática dos periféricos, pela UCP para verificar se algum deles precisa de ser atendido. Este mecanismo peca pelo facto de desperdiçar tempo da UCP e também demorar a atender os periféricos.
- Um método eficaz para a inicialização do processo de transferência é permitir que o periférico indique directamente ao processador que está pronto a transferir dados, activando uma interrupção.
- O processador pode então temporariamente suspender a operação presente, confirmar a interrupção e aceitar a transferência de dados. No fim da transferência o processador retornará ao programa original no ponto da interrupção.

3.3.1. O processo da Interrupção

- Para inicializar a transferência de dados, o periférico activará o seu flip-flop de serviço de interrupção, que possuí a saída conectada directamente ao terminal de interrupção do processador. O flip-flop armazena o sinal de interrupção na sua saída até o tempo em que o processador confirmará e efectuará o *clear* do flip-flop.
- Um sistema simples de interrupção operando nos princípios descritos é mostrado a seguir.

3.3.1. O processo da Interrupção

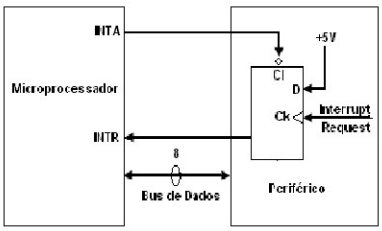


Fig 3.2. Sistema simples de Interrupção

3.3.1. O processo da Interrupção

- Uma vez que a requisição da interrupção pode ocorrer em qualquer instante é assíncrona.
- Ao receber o sinal de interrupção, o processador completa a execução da instrução corrente, confirma a interrupção e automaticamente grava o conteúdo do *program counter* no *stack*. O processador entra agora na rotina de serviço à interrupção, que pode ser vista como uma sub-rotina iniciada externamente.

3.3.2. Interrupções Mascarável e Não-mascarável

- Existem dois tipos de interrupção em uso presente:
  - ***A Interrupção Não-Mascarável (NMI) , e***
  - ***A Interrupção Mascarável (INT)*** .
- Os dois tipos de interrupção são alimentados à uma porta OR, como mostra a figura seguinte, onde são combinados para gerar um único sinal de interrupção mestre. A interrupção não-mascarável é alimentada directamente para esta porta enquanto que a interrupção mascarável através de uma porta AND habilitadora cuja saída é controlada pelo flip-flop habilitador da interrupção.

3.3.2. Interrupções Mascarável e Não-mascarável

Este flip-flop é controlado por sinais inicializados pelo programa. O sinal *enable interrupt* é gerado por escrever a instrução EI no programa e *disable interrupt* é gerado por DI.

Ambas interrupções são normalmente activas em *Low*. A NMI é activa no flanco descendente, enquanto que INT é activa no nível *Low*.

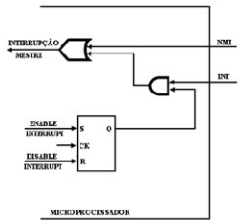


Fig. 3.3. NMI e INT

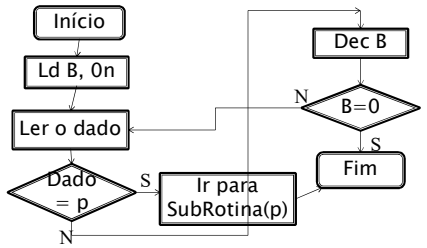
3.3.3. Identificação da fonte de interrupção

- Quando o processador é usado em conjunção com um número de periféricos, uma das suas funções é identificar a fonte de interrupção. Alguns processadores não tem mais que uma linha de interrupção; o Z80 possuí duas linhas.
- Nesse casos várias interrupções são combinadas numa porta OR para formar uma interrupção mestre que subsequentemente é alimentada ao pino de interrupção do processador.

3.3.3. Identificação da fonte de interrupção

- Existem duas técnicas comummente usadas na identificação da fonte de interrupção mestre, nomeadamente *Polling* e *Vectoring*.
- *Polling* é um método de identificação usando software, enquanto que *vectoring* corresponde à aproximação correspondente em hardware.
- Na técnica *Polling*, o periférico, ao requerer a interrupção, deve colocar em simultâneo um código no barramento de dados que será usado pelo processador na identificação da fonte.
- É necessário estabelecer prioridade, e essa depende da ordem em que serão testados os códigos no programa.
- Seja ***n*** o número de periféricos e ***p*** o código introduzido no porto pelo periférico; segue-se o seguinte fluxograma:

3.3.3. Identificação da fonte de interrupção



- *Fig. 3.4. Software polling*

3.3.3. Identificação da fonte de interrupção

- Na técnica *Vectoring*, a identificação da fonte de interrupção pode ser feita usando a ligação *Daisy-Chain*, (figura no diapositivo a seguir).
- Depois da interrupção gerada na saída da OR, o processador responde com o sinal INTA (*Interrupt acknowlegde*) que é propagado do periférico mais prioritário ao menos.
- Quando a fonte da interrupção é detectada, o sinal é bloqueado e não se propaga para os próximos estágios.

3.3.3. Identificação da fonte de interrupção

O sinal DE(*data enable*) é usado para habilitar os bufferes tri-state associados às suas linhas de dados, entregando assim o endereço da rotina de interrupção correspondente.

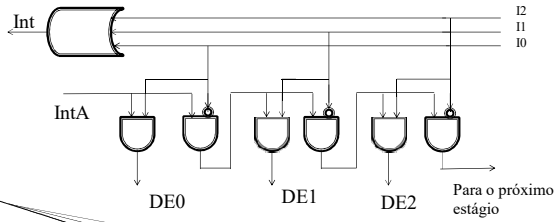


Fig. 3.4. Daisy chain