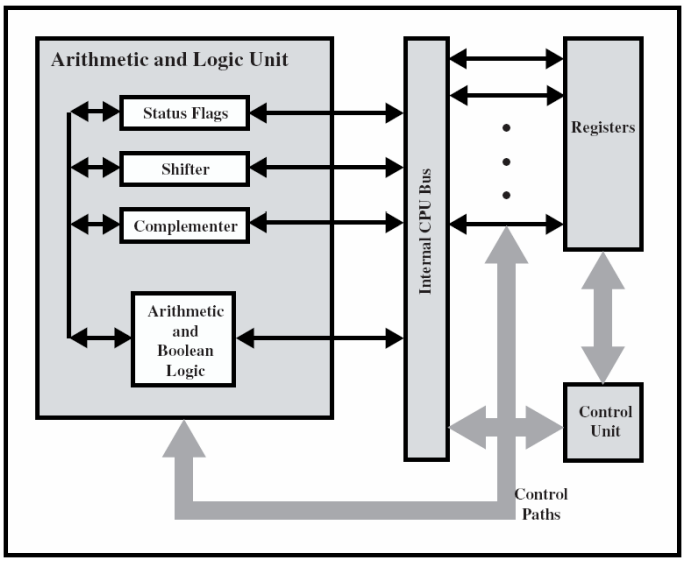
Endereçamento de Memória



Para se entender o funcionamento da CPU deve-se considerar as ações que ela executa:

1. **Busca de instrução**: uma nova instrução é lida da memória.
2. **Interpretação da instrução**: a instrução é decodificada para determinar a ação requerida.
3. **Busca de dados**: a instrução que está sendo executada requer a leitura de dados da memória ou de um módulo de E/S.
4. **Processamento de dados**: a instrução pode requerer a execução de uma operação lógica ou aritmética sobre os dados.
5. **Escrita de dados**: instrução pode requerer que o resultado da execução seja escrito em memória ou em um módulo de E/S.

Organização dos Registradores

Os registradores da CPU exercem duas funções:

1. **Registradores visíveis para o usuário**: utilizado pelos programadores de linguagem de montagem para otimizar o código.
2. **Registradores de controle e de estado:** utilizados pela UC para controlar as operações da CPU.

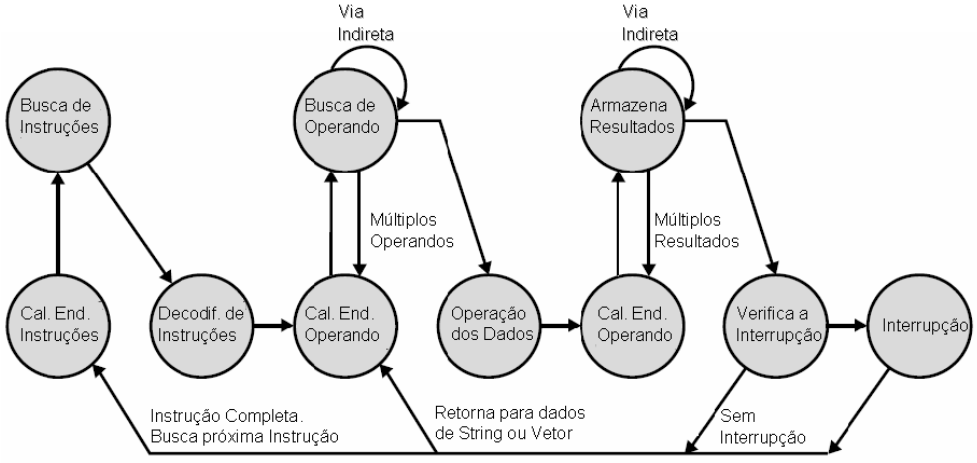
Os **registradores visíveis** para o usuário podem ser classificados em:

1. ***Registradores de propósito geral****:* podem ser utilizados pelos programadores para uma variedade de funções, por exemplo armazenar um operando para qualquer operação.
2. ***Registradores de dados****:* são utilizados apenas para armazenar dados e não podem ser empregados no cálculo de endereços de operandos.
3. ***Registradores de endereço****:* utilizados para um determinado modo de endereçamento (imediato, direto, indireto, etc).
4. ***Registradores de código de condição*** *(conhecidos como flags):* armazenam bits que indicam o resultado de uma operação. Por exemplo, em uma operação aritmética este bit pode indicar se o resultado é positivo, negativo, zero ou *overflow*.

Os **registradores de controle e de estados** são empregados para controlar as operações da CPU. Os seguintes registradores são essenciais para a execução das instruções:

1. *Contador de programa – PC:* armazena o endereço da instrução a ser buscada.
2. *Registrador de instrução –IR:* armazena a última instrução buscada.
3. *Registrador de endereçamento à memória – MAR:* armazena o endereço de uma posição de memória.
4. *Registrador de armazenamento temporário de dados - MBR:* armazena uma palavra de dados a ser escrita na memória ou a palavra lida mais recentemente.

A sequência correta dos eventos envolvidos no ciclo de instrução pode variar de processador para processador, dependendo do projeto desenvolvido. No entanto, de uma forma geral os eventos acontecem conforme ilustra a figura a seguir.



Ciclo de instrução detalhado (com interrupções).

Para exemplificar, considere que uma CPU utilize os seguintes registradores: MAR, MBR, PC e IR. No *ciclo de busca*, uma nova instrução é lida da memória.

O PC contém o endereço da próxima instrução a ser executada. Este endereço será colocado no MAR e posteriormente no barramento de endereço. Suponha que a operação solicitada pela UC seja uma leitura de instrução.

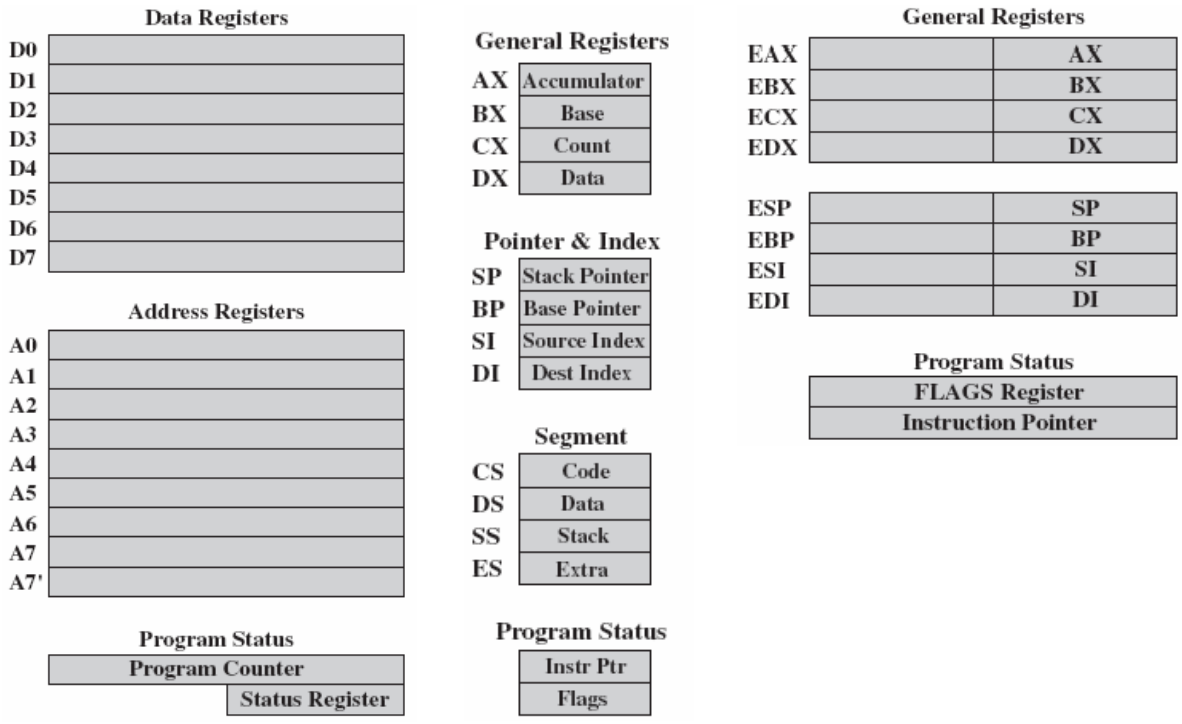
O valor lido na memória (no enderço especificado) é colocado no barramento de dados e depois copiado para dentro do MBR. Por último, o conteúdo de MBR será colocado no IR.

Nesse mesmo instante, o contador de programa (PC) é incrementado em uma unidade, indicando qual é a próxima instrução a ser executada. Devemos ressaltar que dentro de uma mesma instrução mais de uma operação pode ser requisitada.

Registros

As operações do processador envolvem principalmente o processamento de dados. Estes dados podem ser armazenados na memória e acedidos a partir daí. No entanto, a leitura e armazenamento de dados em memória retarda o processador, uma vez que envolve processos complicados de enviar o pedido através do barramento e para a unidade de armazenamento de memória e obter os dados através do mesmo canal.

Para acelerar as operações do processador, o processador inclui alguns locais de armazenamento de memória internos, chamados **registros**.



MC 68000 **8086** **80386 – Pentium 4**

Os registros armazenam elementos de dados para o processador sem ter que aceder à memória. Um número limitado de registros estão incorporados ao chip do processador.

Registros do Processador

Existem dez registros de 32-bit e seis de 16-bit na arquitetura IA-32. Os registros estão agrupados tem três categorias:

- Registros de uso geral,
- Registros de controlo, e
- Registros de segmento.

Os registros de processamento geral ainda estão divididos nos grupos:

- Registros de dados,
- Registros de ponteiro, e
- Registros de índice.

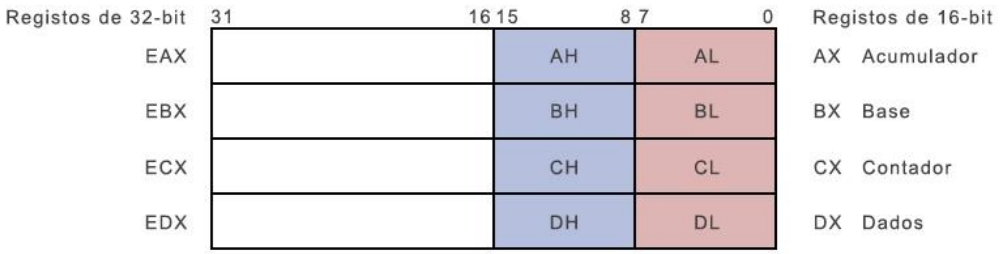
Registros de Dados

Quatro registros de dados de 32-bit são usados para operações aritméticas, lógicas, e para outras operações.

Esses registros podem ser usados de três maneira:

- Como registros completos de 32-bit:
  EAX, EBX, ECX, EDX.
- As metades inferiores dos registros de 32-bit podem ser usados como quatro registros de 16-bit:
  AX, BX, CX, DX.

- As metades mais baixas e mais altas dos registros de 16-bit falamos acima podem ser usados para armazenar dados de 8 bit:
  AH, AL, BH, BL, CH, CL, DH e DL.



Alguns destes registros têm usos específicos em algumas operações aritméticas.

- **O AX é o acumulador primário**: Ele é usado no input/output na maioria das operações aritméticas. Por exemplo, na operação de multiplicação, um operando é armazenado no registro EAX ou AX ou AL de acordo com o tamanho do operando.
- **O BX é conhecido como o registro de base**: Uma vez que pode ser utilizado endereçar índices.
- **O CX é conhecido como o registro de contagem**: Tanto o ECX como o CX, eles armazenam o contador em operações iterativas.
- **O DX é conhecido como o registro de dados**: Também é utilizado em operações de input/output. Ele também é usado com o registro AX juntamente com o DX para operações de multiplicação e divisão envolvendo grandes valores.

Registros Ponteiros

Os registros ponteiros são de registros de 32-bit EIP, ESP e EBP e os seus correspondentes em 16-bit são IP, SP e BP. Existem três categorias de registros apontadores:

- **Instruction Pointer (IP):** Este registro contém o 16-bit do endereço de offset (deslocamento) da próxima instrução a ser executada. O IP em associação com o registro CS (como CS:IP) dá o endereço completo da instrução no segmento de código.
- **Stack Pointer (SP):** O registro SP de 16-bit fornece o valor de deslocamento dentro da *stack* do programa. O SP em associação com o registro SS (SS:SP) refere-se a posição atual dos dados ou do endereço no interior da *stack* do programa.
- **Base Pointer (BP)**: O registro de 16-bit BP ajuda principalmente a referenciar as variáveis passados por parâmetro a uma sub-rotina. O Endereço no registro SS combinado com o *offset* em BP obtém-se a localização do parâmetro. O BP também pode ser combinado com o DI e o SI como base para operações de endereçamento especiais.

Registros de Índice

Os registros de índice de 32-bit são o ESI e o EDI, os seus 16-bit mais à direita são o SI e o DI. O Si e o DI, são usados endereçamento indexado e às vezes também são usados na adição e subtração.

Há dois conjuntos de ponteiros de índice:

- **Source Index (SI):** Ele é usado como índice da fonte de dados a copiar.
- **Destination Index (DI):** Ele é usado como índice do destino dos dados a copiar.

Registros de Controle

Quando se combina os 32-bit dos registros de ponteiros com 32-bit dos registros de *flags* obtém-se os registros de controlo.

Muitas instruções envolvem comparações e cálculos matemáticos, alteração de estados das *flags* e outras instruções condicionais testam os valores do estado dessas *flags* para direcionar o fluxo de execução para outro local.

As *flags* mais comuns são:

- **Overflow Flag (OF):** Ela indica se ocorreu um *overflow* de um bit mais significativo dos dados após uma dada operação aritmética.
- **Direction Flag (DF):** Ela determina em que direção uma *string* é comparada. Se o seu valor for definido para 0, a operação de comparação irá seguir uma direção da esquerda para a direita Quando definido para 1, a comparação irá ser efetuada da direita para a esquerda.
- **Interrupt Flag (IF)**: Esta *flag* define se as interrupções externas, tais como o teclado, serão processadas ou ignoradas. O estado 0 corresponde a ignorar as interrupções externas e ativado quando definida a 1.
- **Trap Flag (TF)**: Ela permite definir o modo de operação do processador para *single-step*. Isto permite que um programa de *DEBUG* ative esta *flag* e assim é possível avançar a execução através de uma instrução de cada vez.
- **Sign Flag (SF):** Ela mostra o sinal do resultado de uma operação aritmética. Esta *flag* é definido de acordo com o sinal dos dados de uma operação aritmética. O sinal é indicado pela bit mais significativo. Um resultado positivo limpa o valor de SF a 0 e um resultado negativo define -o como 1.
- **Zero Flag (ZF):** Ela indica o resultado de uma operação aritmética ou de comparação. Um resultado diferente de zero limpa a *flag* a 0, e um resultado a zero define-a para 1.
- **Auxiliary Carry Flag (AF):** Ela contém o *carry* (transporte) do bit 3 para o bit 4 de uma operação aritmética; usada para operação aritméticas especificas. O AF é definido quando uma operação aritmética de 1 byte provoca um *carry* do bit 3 para o bit 4.
- **Parity Flag (PF):** Ela indica o número total de bits no resultado obtido a partir de uma operação aritmética. Um número par de 1-bit limpa a *flag* para 0 e um número ímpar do 1-bit define a *flag* a 1.
- **Carry Flag (CF):** Contém o *carry* de 0 ou 1 do bit mais significativo após uma operação aritmética. Ela também armazena o conteúdo do último bit de uma operação de *shift* ou de rotação.

A tabela indica a posição dos bits das *flags* no registro de *Flags* em 16-bit:

Flag:					**O**	**D**	**I**	**T**	**S**	**Z**		**A**		**P**		**C**
Bit n.:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Registros de Segmento

Os segmentos são áreas especificas da memória em um programa para conter dados, código e a *stack*.

Existem três principais segmentos:

- **Code Segment:** Este segmento contém todas as instruções a serem executadas. Um registro de código de 16-bit ou registro CS armazena o endereço do início do segmento de código.
- **Data Segment**: Este contém os dados, constantes e áreas de trabalho. Um registro do segmento de dados ou DS armazena o endereço do início do segmento de dados.
- **Stack Segment:** Este contém dados e os endereços de retorno de um procedimento ou sub-rotina. Este é implementado como uma estrutura de dados do tipo *stack*. O registro do segmento da *stack* ou SS armazena o endereço de início do segmento da *stack*.

A parte dos registros DS, CS e SS, existem outros registros extra – ES (Extra Segment), FS, GS, que fornecem segmentos adicionais para armazenar dados.

Na programação *assembly*, o programa precisa de aceder aos locais de memória. Todos os locais de memória dentro de um segmento são em relação ao endereço de início do segmento. Um segmento começa em um endereço divisível por 16 ou pelo hexadecimal 10. Assim, o digito mais à direita em todos estes endereços de memória é 0, que por norma não é armazenado nos registros do segmento.

Os segmentos do registro armazena os endereços de início de um segmento. Para obter a localização exata dos dados ou instrução dentro de um segmento, um valor de *offset* (ou deslocamento) é necessário. Para fazer referência a qualquer posição de memória em um segmento, o processador combina o endereço do segmento no registro com o valor do *offset* da localização.

Modo	Algoritmo	Vantagem	Desvantagem
Imediato	**Op = Ac**	Nenhuma Referência a Memória	Limitada a Magnitude do Operando
Direto	**EA = Ac**	Simples	Espaço de Endereçamento Limitado
Indireto	**EA = (Ac)**	Espaço de Endereçamento Grande	Múltiplas Referências a Memória
Registrador	**EA = R**	Nenhuma Referência à Memória	Espaço de Endereçamento Limitado
Indireto via Registrador	**EA = (R)**	Espaço de Endereçamento Grande	Referência Extra a Memória
Deslocamento	**EA = Ac + (R)**	Flexibilidade	Complexidade
Pilha	**EA = Topo da Pilha**	Nenhuma Referência à Memória	Aplicabilidade Imediata

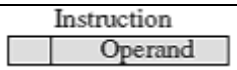
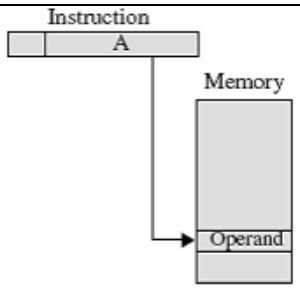
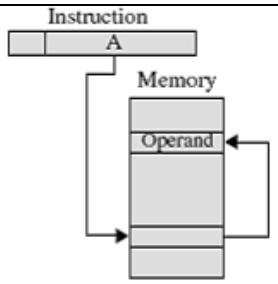
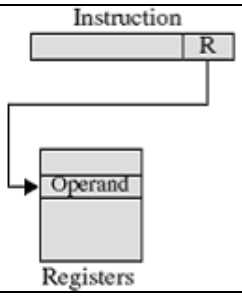
No Quadro acima é utilizado a seguinte notação:

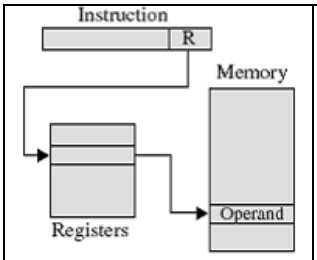
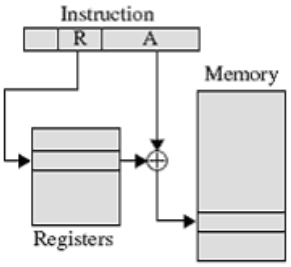
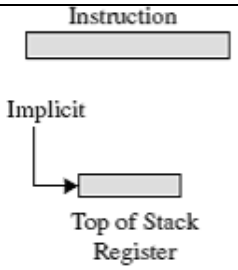
Código	Representa
Ac =	Conteúdo de campo de endereço da instrução
R =	Conteúdo de campo de endereço que referencia um registrador
EA =	Endereço real (efetivo) da posição que contém o operando
(X) =	Conteúdo da posição de endereço X

Existem várias técnicas que podem ser aplicadas para realizar o Endereçamento de Posições de Memória (Principal ou Virtual). A ideia é utilizar a menor quantidade de bits possível da instrução para armazenar os endereços. Os modos mais comuns são:

1. Endereçamento imediato
2. Endereçamento direto
3. Endereçamento indireto
4. Endereçamento de registrador
5. Endereçamento indireto via registrador
6. Endereçamento por deslocamento
7. Endereçamento por pilha

Representação gráfica dos Modos de Endereçamento

			
End. Imediato	End. Direto	End. Indireto	End. De Registrador

		
End. Indireto via Registrador	End. por Deslocamento	End. por Pilha