Sistemas de Informação
Disciplina: Arquitetura e Organização de Computadores - 1º Período
Professor: José Maurício S. Pinheiro

AULA 5: Unidade de Controle

A Unidade Lógica e Aritmética (ULA) é responsável por executar o processamento do computador, mas a ULA realiza apenas operações individuais. Para que a ULA processe sequências de instruções, é necessário que algum dispositivo forneça tais instruções, na ordem correta.

1. Unidade de Controle

Enquanto a ULA é como uma calculadora simples, que executa um pequeno número de operações, a UC é como o operador da calculadora, que sabe onde buscar informações para alimentar a calculadora e também em que ordem estas informações devem ser repassadas. Em outras palavras, enquanto a ULA faz "partes" do trabalho, a UC gerencia a execução destas partes, de forma que um trabalho mais complexo seja executado. Algumas das tarefas da UC são:

- Controlar a execução de instruções na ordem correta: uma vez que a ULA só cuida de executar instruções individuais, a UC tem o papel de ir buscar a próxima instrução e trazê-la para a ULA, no momento correto;
- Leitura da memória principal: a ULA não pode acessar diretamente a memória principal da máquina; realiza somente operações sobre os registradores, sendo que as instruções devem ser comandadas diretamente a ela. A UC tem o papel de buscar as instruções na memória e também verificar se a instrução exige dados que estejam na memória. Se for o caso, a UC deve recuperar os dados na memória e colocá-los em registradores especiais e, finalmente, solicitar que ULA execute a operação sobre estes valores;
- Escrita na memória principal: a ULA também não pode escrever na memória principal da máquina. Quando for necessário armazenar o resultado de uma operação na memória principal, é tarefa da UC transferir a informação do registrador para a memória;
- Controlar os ciclos de interrupção: sinais de interrupção são aqueles que indicam para a UC que ela deve parar, momentaneamente, o que está fazendo e ir executar uma outra tarefa. As razões para as interrupções são as mais diversas, como o disparo de um timer ou uma placa de rede solicitando o descarregamento de seu *buffer*.

Não é a Unidade de Controle quem executa as operações, ela é a responsável apenas por buscar e decodificar as instruções e, a partir daí, gerar os sinais de controle que ativarão as demais partes dor processador. Ou seja, o processamento de dados propriamente dito é feito exclusivamente pela ULA, a qual processa os dados normalmente armazenados nos registradores. A Figura 1 apresenta um diagrama em blocos da unidade de Controle.

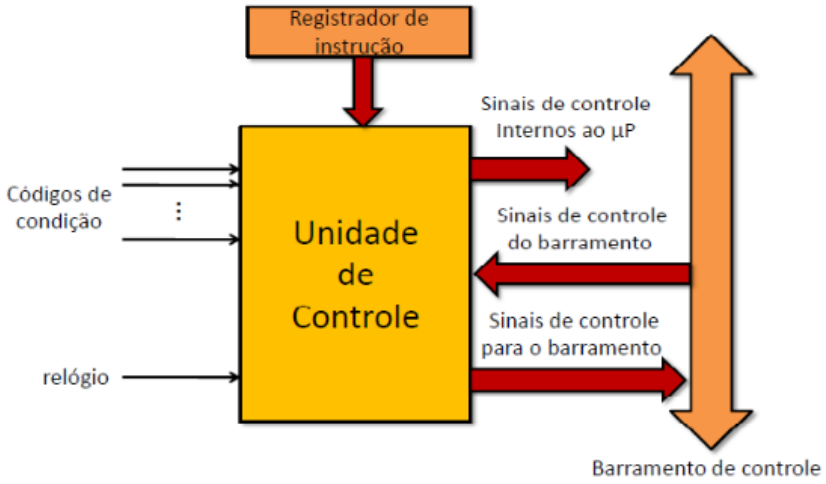

Figura 1 – Diagrama de uma UC

2. Rotina de Operação da CPU

É possível afirmar que a CPU tem uma sequência de ações a executar, algumas são atividades da ULA, outras da UC. Esta sequência é apresentada a seguir:

1. Busca de instrução: quando a CPU lê uma instrução na memória;
2. Interpretação de instrução: quando a CPU decodifica a instrução para saber quais os passos seguintes necessários;
3. Busca de dados: caso seja determinado na interpretação que dados da memória ou periféricos são necessários, a CPU busca estes dados e os coloca em registradores;
4. Processamento de dados: quando a instrução requer uma operação lógica ou aritmética, ela é executada neste instante;
5. Escrita de dados: se o resultado da execução exigir uma escrita na memória ou periféricos, a CPU transfere o valor do registrador para o destino final;
6. Avaliação de interrupções: após finalizar a execução de uma instrução, a CPU verifica se foi requisitada uma interrupção (*Interrupt Request*). Se sim, toma as providências necessárias. Se não, volta para o item 1.

Pelas responsabilidades da ULA e da UC, é possível perceber que a atividade do item 4 é executada pela ULA e todas as outras pela UC.

3. Registradores Usados pela UC

Assim como a ULA tem seus registradores especiais (acumulador para armazenar os resultados e *flags* para indicar informações sobre a última operação executada), a UC precisa de alguns registradores para funcionar corretamente. O primeiro deles vem da necessidade da UC saber onde está a próxima instrução a ser executada. Em outras palavras, ela precisa de um

registrador que indique a posição de memória em que a próxima instrução do programa estará armazenada (para que ela possa realizar a busca de instrução).
Este registrador está presente em todos os computadores, mas seu nome varia de uma arquitetura para outra. Normalmente este registrador é chamado de Contador de Programa (*Program Counter* - PC). Sempre que é iniciado um ciclo de processamento a UC busca a próxima instrução na memória, na posição indicada pelo PC. Em seguida, o PC é atualizado para apontar para a próxima posição da memória (logo após a instrução), que deve indicar a instrução seguinte. A UC precisa analisar esta instrução antes de decidir o que fazer em seguida. Por esta razão deve existir um registrador especial para armazenar a última instrução lida, chamado Registrador de Instruções (*Instruction Register* - IR).
Para conseguir ler e escrever dados em memórias e periféricos, a UC também precisa de um contato com o barramento, o que é feito através de registradores de armazenamento temporário, chamados MAR (*Memory Address Register*) ou Registro de Endereço de Memória e o MBR (*Memory Buffer Register*) ou Registro de Buffer de Memória. Assim, quando é preciso escrever na memória (ou em um periférico), a UC coloca o endereço no registrador MAR, o dado no registrador MBR e comanda a transferência pelo barramento de controle. Quando for preciso ler da memória (ou do periférico), a UC coloca o endereço no MAR, comanda a leitura pelo barramento de controle e então recupera o valor lido pelo MBR.
Adicionalmente a estes registradores, as CPU's costumam ter outros registradores que podem facilitar sua operação e mesmo sua programação. Alguns destes são os registradores de propósito geral, que servem para armazenar resultados intermediários de processamento, evitando a necessidade de muitas escritas e leituras da memória quando várias operações precisarem ser executadas em sequência, a fim de transformar os dados de entrada nos dados de saída desejados.
Existem também os registradores de pilha, normalmente com nomes como *Stack Pointer* (SP), ou Ponteiro da Pilha, que servem para que uma pilha seja usada pelo processador, na memória. Em essência, é onde o endereço de retorno é armazenado, quando um desvio é feito em linguagem de máquina (é preciso saber para onde voltar após a realização de uma chamada de sub-rotina). A pilha que o processador fornece pode ser usada com outros objetivos, como passagem de parâmetros, por exemplo.
Quase todas as arquiteturas fornecem os registradores de índices, que são registradores que permitem acessar, por exemplo, posições de uma matriz. Ele guarda uma posição de memória específica e existem instruções que permitem acessar o *n*-ésimo elemento a partir daquela posição. Seus nomes variam muito de uma arquitetura para outra, como IX, de IndeX, SI, de Source Index (Índice Fonte) ou ainda DI, de Destination Index (Índice Destino).
Arquiteturas com segmento possuem ainda os registradores de segmento, que definem o endereço "zero" da memória para um determinado tipo de informação. A arquitetura x86, por exemplo, possui diversos registradores deste tipo: CS, de Code Segment (Segmento de Código), DS, de Data Segment (Segmento de Dados), SS, de Stack Segment (Segmento de Pilha) e ES, de Extra data Segment (Segmento de Dados Extra). Quando estes

segmentos existem, os endereços usados nos índices, contador de programa e outros são "somados" com os endereços do segmento para que a posição real na memória seja calculada. Por exemplo:

SS = 10000h => Endereço do segmento da pilha
SP = 1500h => Endereço da pilha (dentro do segmento)
Endereço real da pilha = SS + SP = 10000h + 1500h = 11500h

Isso permite que um programa possa rodar em qualquer parte da memória, mesmo que ele tenha sido criado para ser executado no endereço 0h: basta indicar no registrador CS o endereço inicial de carregamento deste programa e, para todas as instruções deste programa, vai ser como se ele estivesse no endereço 0h.
A CPU, com os registradores de segmento, é responsável pela tradução do endereço "virtual" para o endereço real. Vale lembrar que cada arquitetura tem nomes distintos para estes registradores e alguns destes registradores até existem em algumas arquiteturas, mas não são visíveis para o programador, isto é, o registrador está lá e é usado pela CPU, mas o programador não tem acesso direto a eles (embora muitas vezes tenha acesso indireto, como sempre ocorre com os registradores MAR e MBR).

4. Organização da Pilha

Uma pilha é um dispositivo de armazenamento que guarda as informações em uma disposição último a entrar, primeiro a sair (last-in, first-out – LIFO). As pilhas possuem basicamente duas operações: *push*, processo de inserção de dados para armazenamento no topo da pilha (Figura 2) e *pop*, procedimento de remoção dos dados da memória na parte superior da pilha (Figura 3).

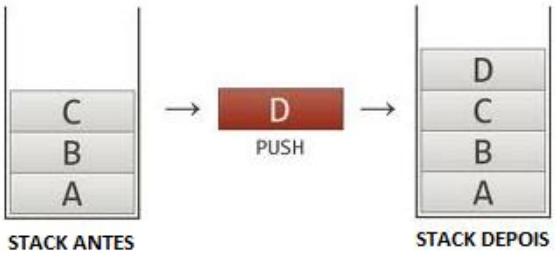

Figura 2 – Operação de PUSH

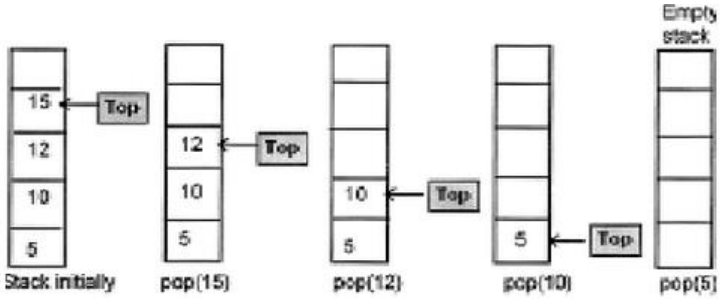

Figura 3 - Operação de POP

Um computador pode ter uma memória separada, reservada apenas para operações de pilha. Entretanto, a maioria utiliza a memória principal para representar pilhas. Assim, todos os programas de montagem devem alocar memória para a utilização de uma pilha. O diagrama de blocos de uma pilha de 64 palavras pode ser visto na Figura 2:

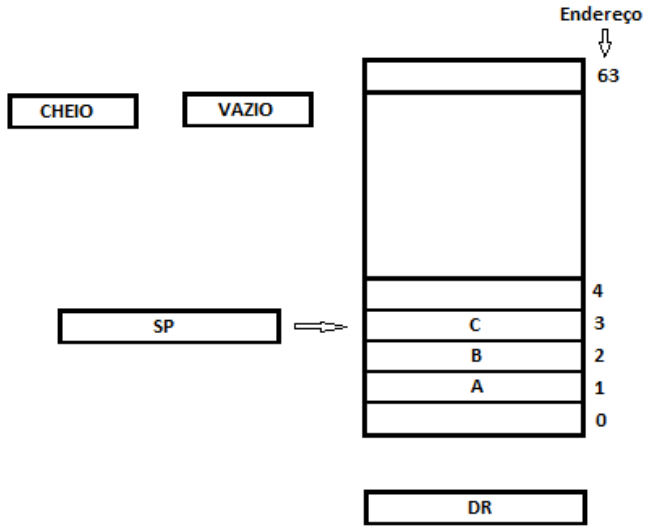

Figura 4 - Diagrama de blocos para uma pilha de 64 palavras

- O registrador SP (*Stack Pointer*) inicialmente é carregado com o endereço do topo da pilha;
- Na memória, a pilha opera de cima para baixo, de modo que quando alguma informação é inserida na pilha, o SP é decrementado.

$$SP \leftarrow SP - 1$$
$$M[SP] \leftarrow DR$$

- Quando algum dado é retirado da pilha, o SP é incrementado:

$$DR \leftarrow M[SP]$$
$$SP \leftarrow SP + 1$$

- As instruções *push* e *pop* podem ser explicitamente executadas em um programa. Entretanto, elas também são implicitamente executadas em outras operações, tais como *procedure calls* (chamadas de procedimentos) e interrupções.

Qualquer tentativa de inserir um novo elemento na pilha que já está cheia resulta em estouro de pilha (Stack Overflow), assim como qualquer tentativa de excluir um elemento a partir dos resultados da pilha já vazia (Stack Underflow). Deve-se tomar cuidado ao efetuar operações de pilha para garantir que um overflow ou underflow da pilha (isto é, a ultrapassagem dos seus limites, para cima ou para baixo) não ocorra.

5. Pilha da Memória

A Figura 5 mostra uma porção de uma memória de computador particionada em três segmentos: Programa, dados e pilha.

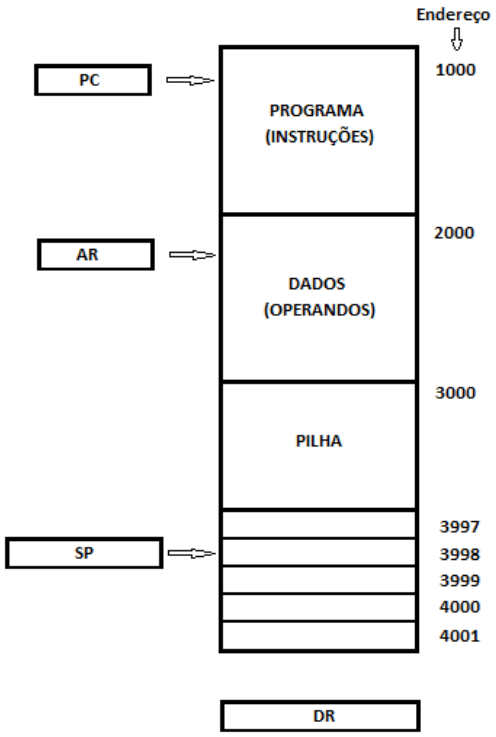

Figura 5 – Memória com os segmentos de programa, dados e pilha

- O contador de programa (PC) aponta para o endereço da próxima instrução do programa;
- O registrador de endereço (AR) aponta para um arranjo de dados;
- O ponteiro da pilha SP aponta para o topo da mesma;
- Os três registradores estão conectados a um barramento de endereços comum, e qualquer um pode fornecer um endereço para a memória;
- O PC é usado durante a fase de coleta, para a leitura de uma instrução;
- O AR é usado durante a fase de execução, para a leitura de um operando;
- O SP é usado para inserir ou retirar itens para ou da pilha;
- O valor inicial do SP é 4001 e a pilha cresce com endereços decrescentes. Assim, o primeiro item é armazenado no endereço 4000, o segundo item é armazenado no último endereço que pode ser utilizado pela pilha, no caso, 3000;
- Assume-se que o item na pilha se comunica com o registrador de dados – DR. Um novo item é inserido a partir de uma operação *push*, conforme segue:

$$SP \leftarrow SP - 1$$
$$M[SP] \leftarrow DR$$

- O ponteiro da pilha é decrementado de modo que o mesmo aponte para o endereço da próxima palavra;
- Uma operação de escrita na memória insere a palavra em DR no topo da pilha;
- Um novo item é deletado a partir de uma operação *pop*, conforme segue:

$$DR \leftarrow M[SP]$$
$$SP \leftarrow SP + 1$$

- O item no topo da pilha é transferido para o registrador DR;
- O ponteiro da pilha é então incrementado de modo que o mesmo aponte para o próximo item da mesma.

6. Ciclo de Instrução

Ciclo de instrução é a sequência de passos que a UC segue até que uma instrução seja executada. Os principais subciclos são os de busca de instruções, busca de dados, execução e interrupção conforme mostrado no diagrama da Figura 4.

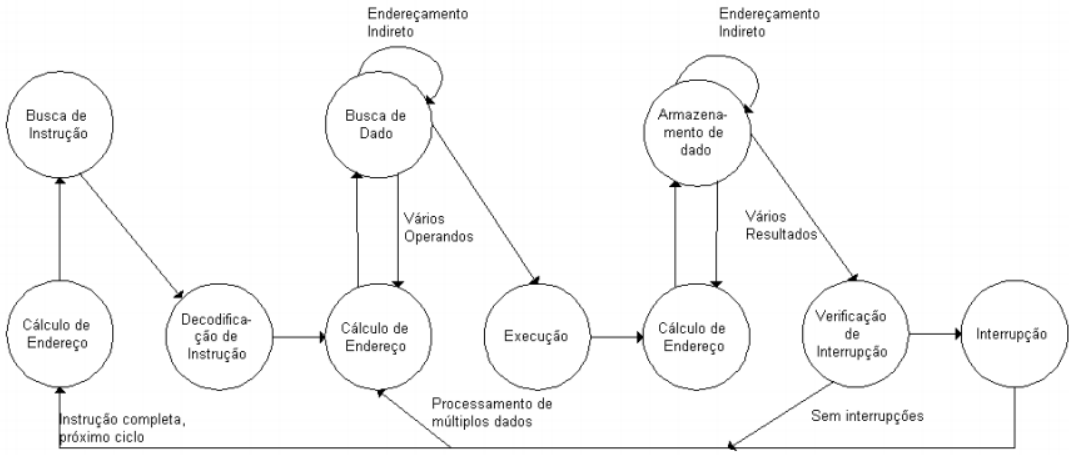

Figura 6 - Diagrama de transição de estados do ciclo de instrução

- O primeiro passo é a busca de instrução, onde a UC coloca o valor do PC no MAR, comanda leitura da memória e recebe o dado (que neste caso é uma instrução) pelo MBR, que em seguida copia para o IR.
- Em seguida, a UC decodifica a instrução, avaliando se há a necessidade de busca de dados adicionais (por exemplo, se for uma instrução do tipo "ADD A,B", nenhum dado precisa ser buscado.
- Se houver a necessidade, o próximo passo é a busca de dado, onde a UC realiza o mesmo processo da leitura da instrução, mas agora para a leitura do dado. Esse processo é repetido até que todos os dados necessários tenham sido colocados em registradores.
- O passo seguinte é a execução, onde a UC comanda a ULA para executar a operação relevante. A ULA devolve um resultado em um registrador.

- Se este dado precisar ser armazenado externamente à CPU, a UC coloca este dado na MBR e o endereço destino na MAR e comanda a escrita, usando o barramento de controle. Se houver mais de um dado a armazenar, este ciclo é repetido.
- Finalmente, a UC verifica se há requisição de interrupção pendente. Se houver, ela executa o ciclo da interrupção (que varia de arquitetura para arquitetura). Caso contrário, o funcionamento prossegue, com o cálculo do novo endereço de instrução e o ciclo recomeça.

Convém salientar que, apesar de os estágios de execução de uma instrução (como a busca de instrução, decodificação de instrução etc.) - parecerem simples, sua implementação com circuitos lógicos é bastante complexa e, em geral, impõe uma grande dificuldade para modificações e expansões no conjunto de instruções.