Programação do 8085

Os projetistas de µP deixam disponível para os programadores um conjunto de instruções de máquina. Estas instruções são sequências binárias que quando decodificadas pelo 8085 vão executar as tarefas a que se propõem. Estas instruções são relativamente simples, como mover dados entre registradores ou fazer uma operação na ULA. As instruções no 8085 podem ter 1, 2 ou 3 bytes, sendo que o primeiro byte sempre indica o que ela faz (OPCODE – código operacional) e os bytes restantes, se houver, são complementos da instrução.

Linguagem de máquina e linguagem *assembly*

Escrever um programa em linguagem de máquina (binário) cria muitas dificuldades na própria escrita e na depuração do programa. Para amenizar esta parte, usa-se a linguagem ***assembly*** que tem uma correspondência de 1 para 1 (1 instrução *assembly* equivale a 1 instrução de máquina) e esta linguagem está baseada no uso de mnemônicos (instruções como ADD, MOV, ADD, etc) para as instruções. Esta linguagem é diferente das linguagens de alto e médio nível como por exemplo a linguagem C. Para cada comando em C pode ser preciso dezenas de comandos de máquina ou *asssembly* para a sua equivalência.

Um programa escrito em linguagem de alto nível pode ser "traduzido" para a linguagem de máquina através de um programa chamado compilador. O **codeblocks**, programa muito utilizado para a produção de programas em C, tem no seu conjunto de programas, um copilador C que transforma o programa-fonte escrito em um arquivo objeto que será processado pelo linkeditor e aí teremos um arquivo executável para aquela plataforma usada (no caso os PCs que você usava durante as aulas de ATP).

Da mesma forma, temos um programa que transforma o programa-fonte em *assembly* para um arquivo executável em linguagem de máquina. Este programa se chama ***assembler*** (montador) e como existe uma correspondência direta entre as instruções *assembly* e de máquina, este programa "trabalha" menos comparado ao compilador.

Execução de um programa no 8085

Os µP são circuitos digitais que buscam uma instrução e a executam, partindo para a busca da próxima instrução e sua execução, e seguem neste ciclo indefinidamente. Lembrando que os registradores de dados (A, B, C, D, E, H, L) têm 8 bits e os que armazenam endereços (PC, SP) têm 16 bits, para que o 8085 não se "perca" na execução, ele precisa ter um marcador indicando em qual instrução ele está no momento. Quem faz esse papel é o PC (*program counter* – contador de programa) que tem 16 bits e armazena o endereço da próxima instrução a ser executada. No 8085 o PC inicia em 0000H quando o microprocessador é alimentado. O 8085 então busca a instrução que está no endereço 0000H e é incrementado, armazenando o novo valor 0001H, ou seja, está sempre

“apontando” para a próxima instrução a ser executada enquanto a instrução atual está sendo executada.

Conjunto de instruções do 8085

As instruções no 8085 são distribuídas em 5 grupos:

1 – Grupo de transferência da dados: Move dados entre registradores ou posição de memória e registradores;

2 – Grupo Aritmético: Adição, subtração, Incrementos, decremento;

3 – Grupo Lógico: AND, OR, XOR, Comparação, Rotação, Complemento;

4 – Grupo de Desvio: Condicionais, Incondicionais, Subrotinas;

5 – Grupo de Controle, Pilha, Entrada e Saída.

Exemplos de algumas instruções

Nas instruções de 1 byte, ele é o próprio OPCODE, não precisando de complementos. Como exemplo temos:

MOV A, B

O mnemônico MOV indica uma movimentação, transferência. As instruções MOV fazem a cópia do conteúdo do registrador fonte (sempre o da direita, neste caso o B) para o registrador destino (sempre o da esquerda, neste caso o A). Ao terminar a execução desta instrução, será escrito no registrador A o conteúdo de B.

ADD B

O mnemônico ADD indica uma adição. Aqui vemos somente um registrador especificado, no caso o B. Lembrando que em uma operação na ULA o registrador A está ligado a uma das entradas da ULA e recebe o resultado dela. Portanto o A é subentendido nestas operações e a soma será feita do conteúdo de A com o conteúdo de B, indo o resultado para o A.

Nas instruções de 2 bytes, temos o OPCODE e um complemento que é um byte. Como exemplo temos:

MVI B, 0

O mnemônico MVI indica movimentação de imediato, sendo que imediado é um número. Portanto ao terminar a execução desta instrução teremos B com o valor 0. Esta instrução faz atribuição de um valor numérico a um registrador.

ANI 25

O mnemônico ANI indica um AND e um imediato. Mais uma vez o registrador A está subentendido na instrução e ao final teremos o A recebendo um AND lógico do valor anterior de A com o valor binário de 25.

As instruções de 3 bytes tem um byte de OPCODE e um complemento de 2 bytes, que indica uma informação de endereço. Como exemplo temos:

LDA 2030H

LDA significa *load* A. Ele carrega o conteúdo do endereço de memória especificado no registrador A. Veja que esta memória é a memória RAM do sistema e se localiza fora do µP

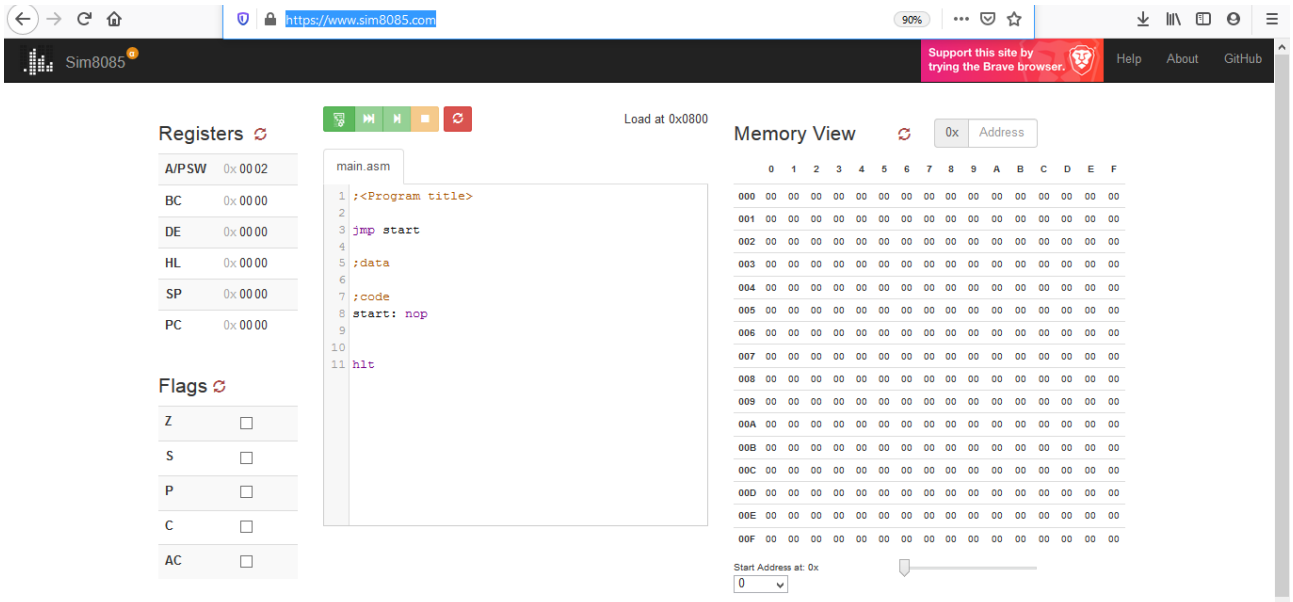
STA 5050H

STA significa *store* A. Ele faz o inverso do LDA, transferindo o conteúdo de A para o endereço de memória especificado na instrução.

Simuladores 8085

Existem simuladores do 8085 para serem instalados ou online. Por questões de facilidade, usaremos o simulador online sim8085, disponível em https://www.sim8085.com .

Na figura abaixo vemos a janela do aplicativo que pode ser executado em qualquer navegador. Temos 3 colunas: Na primeira, temos os valores dos registradores em hexadecimal (veja o 0x antes) e os flags da ULA. Na segunda coluna é onde podemos escrever nosso programa e na última coluna temos a memória que é associada ao sistema microprocessado que ele simula.



Navegador exibindo o simulador 8085 online