O objetivo desta atividade é permitir que o aluno seja capaz de criar um programa em linguagem assembly que utilize instruções lógicas, de deslocamento e de rotação.

Parte 1 – Instruções lógicas

As instruções lógicas em linguagem assembly permitem a manipulação de bits de um registrador ou variável. Elas podem ser utilizadas para setar um bit, resetar um bit ou inverter um bit.

Setar um bit (X -> 1)

Para setar um determinado bit (1), utiliza-se a instrução **OR** e uma máscara com o bit 1 na posição que se deseja setar e o bit 0 nas demais posições.

Suponha, por exemplo, que o registrador AL contém o valor 10000001 e deseja-se setar o 3º bit (da direita para a esquerda). Neste caso, utilizaríamos a instrução OR e a máscara 00000100b (bit 1 na 3ª posição). A instrução utilizada então seria:

OR AL,00000100b

Como resultado, o valor de AL que antes era 10000001 ficaria 10000101.

Resetar um bit (X -> 0)

Para resetar um determinado bit (0), utiliza-se a instrução **AND** e uma máscara com o bit 0 na posição que se deseja resetar e o bit 1 nas demais posições.

Suponha, por exemplo, que o registrador AL contém o valor 10000001 e deseja-se resetar o 1º bit (da direita para a esquerda). Neste caso, utilizaríamos a instrução AND e a máscara 11111110b (bit 0 na 1ª posição). A instrução utilizada então seria:

AND AL,11111110b

Como resultado, o valor de AL que antes era 10000001 ficaria 10000000.

Inverter um bit (0->1 ou 1->0)

Para inverter um determinado bit, utiliza-se a instrução **XOR** e uma máscara com o bit 1 na posição que se deseja inverter e o bit 0 nas demais posições.

Suponha, por exemplo, que o registrador AL contém o valor 10000001 e deseja-se inverter o 1º e o 5º bits (da direita para a esquerda). Neste caso, utilizaríamos a instrução XOR e a máscara 00010001b (bit 1 na 1ª e na 5ª posições). A instrução utilizada então seria:

XOR AL,00010001b

Como resultado, o valor de AL que antes era 10000001 ficaria 10010000.

Podemos utilizar instruções lógicas, por exemplo, para fazer a conversão entre letras maiúsculas e minúsculas. Se observarmos a tabela ASCII, todas as letras maiúsculas possuem o 6º bit igual a 0 e todas as letras minúsculas possuem o 6º bit igual a 1. Esta é a única diferença entre letras maiúsculas e minúsculas.

Para realizar a conversão de uma letra qualquer para minúscula basta setar o 6º bit. Assim, caso a letra seja maiúscula ela será convertida para minúscula.

Ex: “A” (01**0**00001) -> “a” (01**1**00001)

Perceba que caso a letra já seja minúscula, ela permanece inalterada.

Ex: “a” (01**1**00001) -> “a” (01**1**00001)

De maneira análoga, para realizar a conversão de uma letra qualquer para maiúscula basta resetar o 6º bit. O exemplo a seguir ilustra como implementar um programa em assembly que lê uma letra maiúscula do teclado e converte a letra digitada para minúsculo.

Ex:

```
TITLE Converte
.MODEL SMALL
.STACK 100h
.CODE
; Lê uma letra maiúscula do teclado (caracter lido é armazenado em AL)
MOV AH,1
INT 21h
; Converte a letra de maiúsculo para minúsculo (seta o 6º bit)
OR AL,00100000b
; Exibe a letra em minúsculo
MOV AH,2
MOV DL,AL
INT 21h
; Finaliza o programa
MOV AH,4Ch
INT 21h
END
```

Testar um bit (verificar se o bit é 0 ou 1)

Para testar o valor de um determinado bit, utiliza-se a instrução **TEST** e uma máscara com o bit 1 na posição que se deseja testar e o bit 0 nas demais posições.

Suponha, por exemplo, que o registrador AL contém o valor 10000001 e deseja-se testar se o 1º bit (da direita para a esquerda) é 0 ou 1. Neste caso, utilizamos a instrução TEST e a máscara 00000001b (bit 1 na 1ª posição). A instrução TEST realiza um AND e não armazena o resultado, mas modifica o registrador de FLAGs. Se o resultado da instrução TEST for 0, significa que o bit testado é 0. Se o resultado da instrução TEST for diferente de 0, significa que o bit testado é 1. Assim podemos utilizar as instruções JZ (ou JNZ) para tomar uma decisão baseada no resultado da instrução TEST.

Ex:

```
TEST AL,00000001b
JZ EHZERO
   ; Executa este trecho se o 1º bit é um
JMP FIM
EHZERO:
   ; Executa este trecho se o 1º bit é zero
FIM:
```

Parte 2 – Instruções de deslocamento

As instruções de deslocamento em linguagem assembly permitem o deslocamento dos bits de um registrador ou variável para a direita ou esquerda uma ou mais posições.

As instruções SHR (Shift Right) a SAR (Shift Arithmetic Right) realizam o deslocamento para a direita e as instruções SHL (Shift Left) e SAL (Shift Arithmetic Left) realizam o deslocamento para a esquerda.

A quantidade de posições deslocadas também deve ser passada como parâmetro para a instrução. Para deslocar apenas 1 posição deve-se utilizar o valor 1 como parâmetro. Para deslocar mais de uma posição, deve-se copiar a quantidade de posições para o registrador CL e utilizar o registrador CL como parâmetro.

Ex:

```
SHR AL,1      ; desloca os bits de AL 1 posição para a direita

MOV CL,3
SHR AL,CL     ; desloca os bits de AL 3 posições para a direita
```

A diferença da instrução SHR para a instrução SAR é que o SHR copia um bit 0 para as posições vagas, enquanto o SAR copia o bit de sinal (bit mais à esquerda) para as posições vagas, ou seja, a instrução SAR realiza o deslocamento dos bits preservando o sinal do número. Já as instruções SHL e SAL são idênticas, uma vez que ambas copiam um bit 0 para as posições vagas.

O último bit deslocado sempre é copiado para a flag CF do registrador de FLAGs. Assim podemos utilizar a instrução JC (salta se CF=1) ou JNC (salta se CF=0) para testar se a flag CF do registrador de FLAGs é 0 ou 1.

Laboratório de
Organização de Computadores
Atividade 6

Atividade para entrega

Crie um programa em linguagem assembly chamado **ATIV6.ASM** que lê um caracter do teclado e exibe na linha seguinte o código ASCII do caracter lido em binário.

Exemplo:

C:\> ATIV6.EXE

Digite um caracter: **A**

Codigo ASCII: **01000001**

ENTREGA

Cada aluno deve:

1) Criar uma pasta em seu escaninho no AVA com o nome **Atividade6.**
2) Postar o arquivo **ATIV6.ASM** dentro da pasta **Atividade6**.