Faculdade Dom Bosco de Porto Alegre
Bacharelado em Sistemas de Informação
Organização e Arquitetura de Computadores – 2025/II
Prof. Filipo Mór – www.filipomor.com

Estudo Dirigido

Simulador NASM (Assembler 8bits)

Converta os códigos em linguagem C a seguir para programas em Assembly da arquitetura NASM 8bits. Indique os valores de todas as variáveis ao final da execução, mostrando o mapa de memória correspondente.

(a)

```c
int A=4, B=2, C=0;
c = A + B;
```

(b)

```c
int A=4, B=2, C=0;
c = A - B;
```

(c)

```c
int A=4, B=2, C=0;
c = (A + B) * 2;
```

(d)

```c
int A=4, B=2, C=2, D=0;
D = (A + B) * C;
```

(e)

```c
int A=4, B=2, C=0, D=0;
while (D<5)
{
    C += ( A + B );
    D++;
}
```

(f)

```c
int A=4, B=2, C=0, D=0;
for(D=0; D<5; D++)
{
    C += ( A + B );
}
```

(g)

```c
int A=0, B=0, C=0, D=0;
while ( D < 10)
{
    if( D < 5)
    {
        A++;
    }
    else
    {
        B++;
    }
    C += A + B;
    D++;
}
```

(h)

```c
int A[3] = {1, 2, 3};
int B[3] = {3, 2, 1};
int C[3] = {0, 0, 0};
int i=0;
for(i=0; i<3; i++)
{
    C[i] = A[i] + B[i];
}
```

Faculdade Dom Bosco de Porto Alegre
Bacharelado em Sistemas de Informação
Organização e Arquitetura de Computadores – 2025/II
Prof. Filipo Mór – www.filipomor.com

Resumo da Linguagem Assembly – Simulador 8 bits

Este documento apresenta um resumo em português da linguagem assembly suportada pelo simulador 8 bits compatível com NASM,
baseado na documentação do site https://professorfilipo.github.io/assembler-simulator/instruction-set.html .

1. Labels (rótulos)

Um label é um nome simbólico que identifica um endereço de memória ou ponto de código.
Deve terminar com dois-pontos (:).
Pode ser usado em saltos (JMP, JZ, etc.) ou como referência de endereço.

- Exemplo:

```
inicio:
   MOV A, 10
.loop:
   DEC A
   JNZ .loop
   HLT
```

2. Valores numéricos e de texto

Os números podem ser representados em diferentes bases:
- Decimal: 200 ou 200d
- Hexadecimal: 0xA4
- Binário: 101b
- Octal: 0o48

- Exemplo:

```
MOV A, 255
MOV B, 0xF0
```

Textos e caracteres são definidos com a diretiva DB. Cada caractere é armazenado em sequência na memória.

- Exemplo:

```
msg: DB "Hello World!", 0
letra: DB 'A'
```

3. Registradores

O simulador define registradores A, B, C e D, além de SP (stack pointer). Todos são de 8 bits (0–255).
O acesso indireto à memória pode ser feito com colchetes:
[A] → conteúdo da memória no endereço A.
[SP+2] → acesso com deslocamento.

4. Instruções

MOV
Copia um valor de uma fonte para um destino.

- Exemplo:

```
MOV A, B
MOV [100], A
```

ADD / SUB
Soma ou subtrai valores e atualiza as flags C (carry) e Z (zero).

- Exemplo:

```
ADD A, 5
SUB B, [valor]
```

INC / DEC
Incrementa ou decrementa um registrador em 1.

- Exemplo:

```
INC C
DEC D
```

MUL / DIV
Multiplica ou divide valores, usando A como acumulador.

- Exemplo:

```
MUL B
DIV C
```

AND / OR / XOR / NOT
Operações lógicas bit a bit.

- Exemplo:

```
AND A, 0x0F
NOT B
```

SHL / SHR
Deslocamento lógico para esquerda ou direita.

- Exemplo:

```
SHL A, 1
SHR B, 2
```

CMP
Compara dois valores e ajusta as flags para saltos condicionais.

- Exemplo:

```
CMP A, 10
JZ fim
```

JMP
Salta incondicionalmente para um label.

- Exemplo:

```
JMP inicio
```

Jumps condicionais
Executam saltos baseados nas flags (Z e C).

- Exemplo:

```
JZ fim
JNZ loop
JC erro
JNC ok
```

CALL / RET
Chamadas e retornos de sub-rotinas.

- Exemplo:

```
CALL subrotina
...
subrotina:
  INC A
  RET
```

PUSH / POP
Manipulam valores na pilha (stack).

- Exemplo:

```
PUSH A
MOV A, 5
POP A
```

HLT
Interrompe a execução do programa.

- Exemplo:

```
HLT
```

5. Observações importantes

- Memória total: 256 bytes (endereços 0–255).
- Todos os valores são 8 bits sem sinal (0–255).
- Flags: Z (zero) e C (carry) controlam os saltos condicionais.

- A faixa de memória 232–255 é exibida na área de saída (visualização) do simulador.