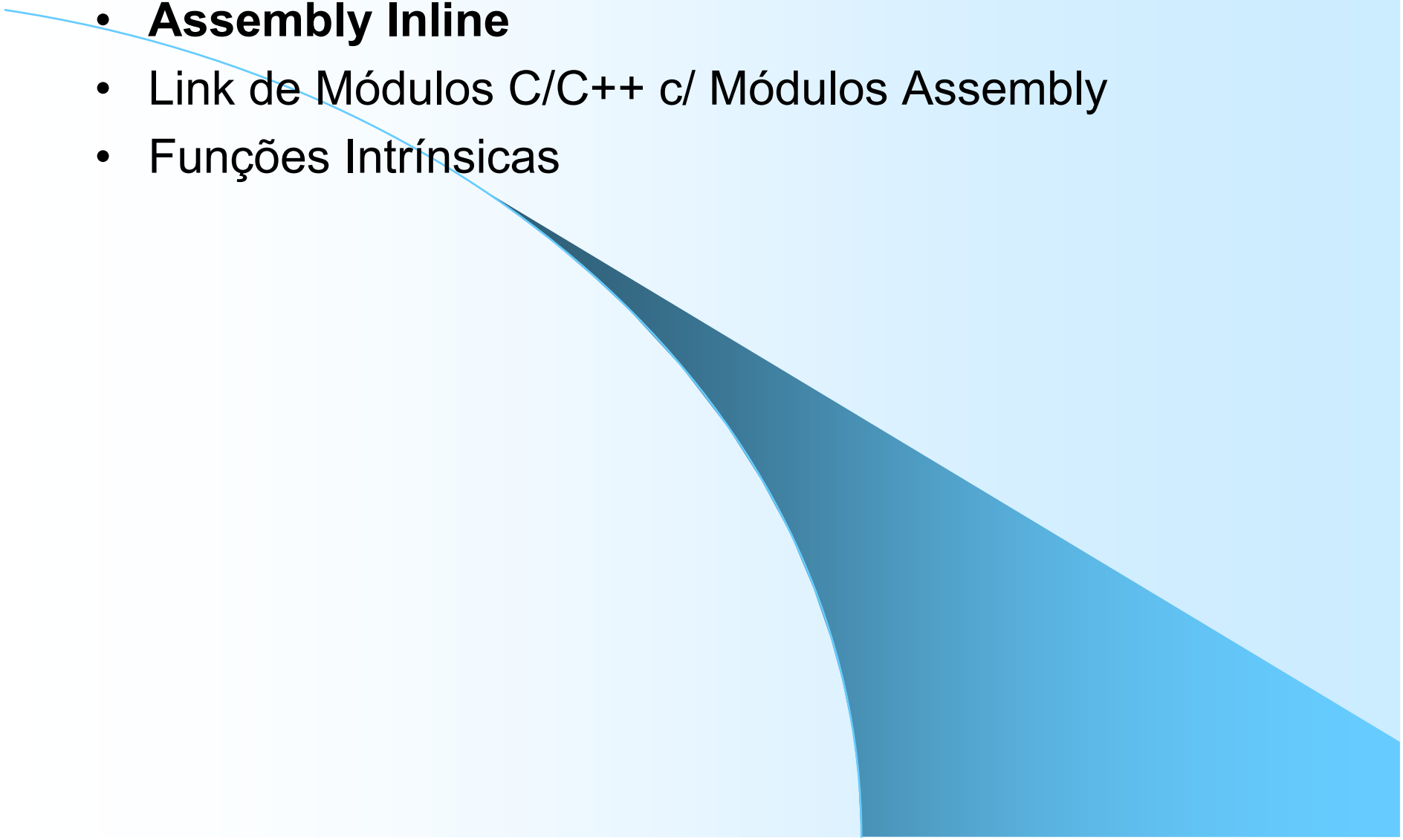
Assembly Language for Intel-Based Computers, 5th Edition

Kip Irvine

Capítulo 12: Interface Assembly-C/C++

Interface Assembly – C/C++

- **Assembly Inline**
- Link de Módulos C/C++ c/ Módulos Assembly
- Funções Intrínsicas



Assembly Inline

- Código Assembly inserido diretamente em um programa escrito em linguagem de alto nível
- Compiladores como VS C++, Borland C++ e GCC fornecem diretivas de compilação p/ identificar e incluir código assembly.
- Código inline executa mais rapidamente porque instruções CALL e RET não são necessárias
- Mais simples de ser implementado porque não existe a necessidade de se preocupar com nomes externos, convenções de nomes, modelos de memória, e passagem de parâmetros.

- ….Porém, não é uma solução portátil.

__asm: Diretiva p/ MS Visual C++

- Uma por instrução, ou demarcando um bloco de instruções
- Syntax:

```
__asm  statement
__asm {
   statement-1
   statement-2
   ...
   statement-n
}
```

Comentários

Qualquer uma das opções abaixo, porém as duas últimas são preferíveis:

```
mov  esi,buf      ; initialize index register
mov  esi,buf     // initialize index register
mov  esi,buf     /* initialize index register */
```

O que pode ser usado em inline assembly

- Qualquer instrução IA-32
- Registradores como operando (eax, ebx, etc)
- Nomes de parâmetro de funções
- Variáveis declaradas fora do bloco __asm
- Rótulos (labels) declaradas fora do bloco __asm
- Literais numéricos usando syntax assembly ou C
- Operador PTR (ex: inc BYTE PTR [esi])
- Diretivas LENGTH, TYPE, SIZE
- Diretivas EVEN e ALIGN

O que NÃO pode ser usado em inline assembly

- Diretivas para definição de dados (ex: DB, DW, BYTE)
- Operadores em geral
- STRUCT, RECORD, WIDTH, and MASK
- Operador OFFSET (como alternativa, usar a instrução LEA)
- Macro diretivas como MACRO, REPT, IRC, IRP
- Referenciar segmentos pelo nome (registradores de segmento: ok)

Uso de Registradores

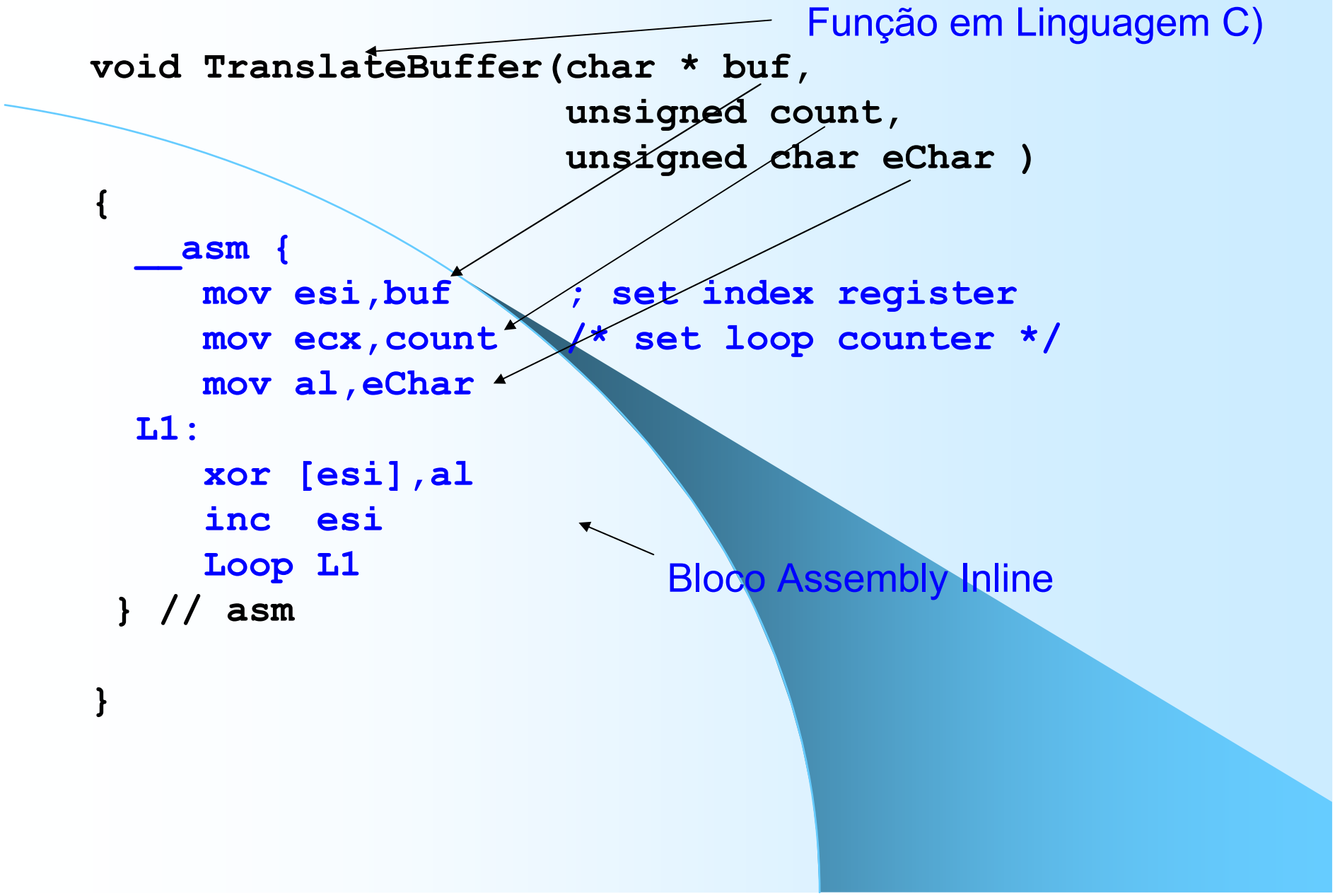
- Em geral, pode-se modificar EAX, EBX, ECX, and EDX no´código inline porque o compilador não assume que esses valores serão preservados entre comandos sucessivos.
- Entretanto, sempre salve e restaure ESI, EDI, e EBP.

Exemplo: Encriptação de Arquivo

- Passos:
  - Ler arquivo do disco
  - Encriptar (usando XOR)
  - Gravar em disco

- Função TranslateBuffer usa um bloc __asm block to implementar comandos que lêem todos os caracters de um vetor, aplicando XOR a cada um deles.

TranslateBuffer



Programa Principal

```
while (!infile.eof() )
  {
    infile.read(buffer, BUFSIZE );
    count = infile.gcount();
    TranslateBuffer(buffer, count, encryptCode);
    outfile.write(buffer, count);
  }
```

Chamada da função C contendo o Bloco Assembly Inline

Programa Principal (versão 2)

```
while (!infile.eof() )
  {
    infile.read(buffer, BUFSIZE );
    count = infile.gcount();
    __asm {
       lea esi,buffer
       mov ecx,count
       mov al, encryptChar
    L1:
       xor [esi],al
       inc  esi
       Loop L1
   } // asm
    outfile.write(buffer, count);
  }
```

Uso do Bloco Bloco Assembly Inline diretamente no código (sem chamar a função)

Por que usar Inline Assembly c/ C/C++ ?

Por que usar Inline Assembly c/ C/C++ ?

- Em geral, usar linguagem de alto nível para desenvolver o projeto

- Usar assembly para:
  - Otimização: Acelerar trechos críticos para o desempenho
  - Acessar dispostivos de hardware não padronizados (ou que não possuam um driver implementado).
  - Escrever código específico para a plataforma em questão
  - Extender os recursos da linguagem de alto nível:
    - Ex: Uso de instruções SIMD

Por que usar Inline Assembly c/ C/C++ ?

- Vantagens:
  - Acesso direto ao hardware
  - Eficiência em tempo de processamento
  - Eficiência em espaço de armazenamento (memória RAM utilizada pelo programa)
- Desvantagens:
  - Reduz a produtividade do programador
  - Alto custo de manutenção
  - Falta de Portabilidade do programa fonte

Otimização de Código

- Regra 90/10: 90% do tempo de CPU de um programa é gasto executando 10% do código
- Logo, otimizações usando ASM devem se concentrar nessa fração de 10%
- Em geral, otimização de loops é a forma mais efetiva de se conseguir ganhos de desempenho.
- Exemplos:
  - Substituir variáveis na memória por registradores, de modo a reduzir o número de acessos à memória
  - Mover código invariante para for a do loop
  - Quando possível, utilizar instruções do tipo SIMD

Inline Assembly – Exemplo de Aplicação

- Projeto Final da Disciplina Laboratório de Arquitetura 2
  - Ver descrição do projeto
  - Discussão em sala de aula e exemplos de implementação
  - Ver material de apoio sobre vetores e matrizes nos próximos slides.

Material Suplementar: Matrizes →

Arrays multidimensionais

- Programadores gostam de pensar em arrays multidimensionais como sendo retângulos (2D) ou cubos (3D)
- The computer stores all multi-dimensional arrays as 1-D arrays
- Porém, o computador armazena todo array multidimensional como um array 1D

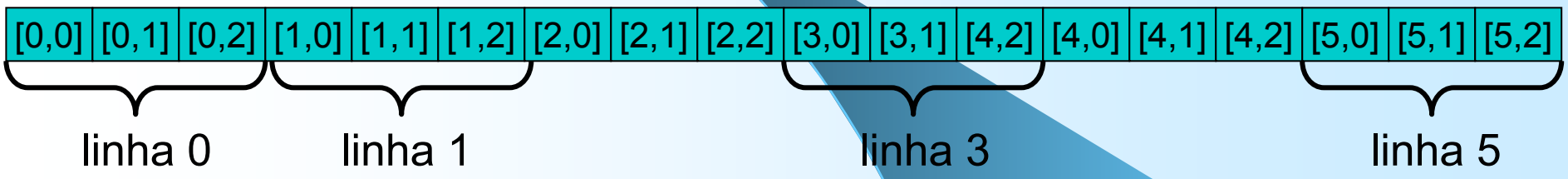
- Ou seja, o array multidimensional é fatiado em linhas (ou colunas), e armazenado sequencialmente na memória.
  - Isso já era esperado pois o endereçamento da memória é sempre unidimensional (sequencia de endereços de bytes).

Arrays 2-D

int x[6][4];

[0,0]	[0,1]	[0,2]
[1,0]	[1,1]	[1,2]
[2,0]	[2,1]	[2,2]
[3,0]	[3,1]	[4,2]
[4,0]	[4,1]	[4,2]
[5,0]	[5,1]	[5,2]

Armazenamento na memória:



endereço de x[i,j] =

endereço de x[0,0] +
i * number of columns * sizeof(int) +
j * sizeof(int)

Exercício

- Considere a seguinte declaração: int x[12][8];
- Assuma que o endereço de x[0,0] é: 004Dh
- Qual é o endereço de x[3,6]?

- Considere a seguinte declaração : char y[32][32];
- Assuma que o endereço de y[0,0] é: 0400h
- Qual é o endereço de y[10,2]?

Exercício

- Considere a seguinte declaração: int x[12][8];
- Assuma que o endereço de x[0,0] é: 004Dh
- Qual é o endereço de x[3,6]?

- endereço de x[3,6] = 004Dh + (3 * 8 * 4)d + (6 * 4)d
  = 004Dh + 96d + 24d
  = 004Dh + 60h + 18h
  = 004Dh + 0078h
  = 00C5h

Exercício

- Considere a seguinte declaração : char y[32][32];
- Assuma que o endereço de y[0,0] é: 0400h
- Qual é o endereço de y[10,2]?

- Endereço de y[10,2] = 0400h + (32 * 1 * 10)d + (2 * 1)d
  = 0400h + 322d
  = 0400h + 0142h
  = 0542h