Espaço de endereçamento de um programa

Acções dos tradutores e ligadores

Endereços virtuais e reais

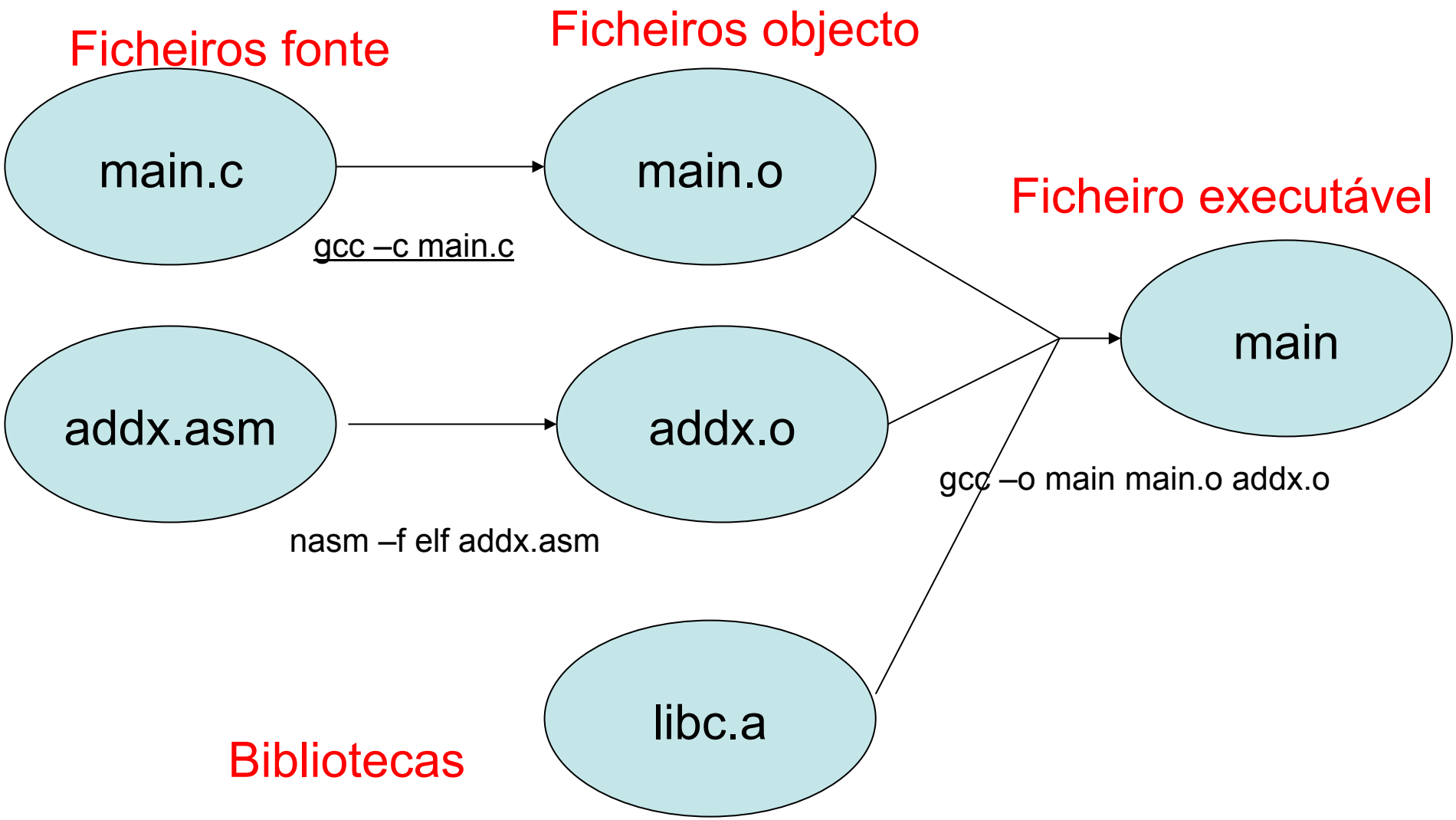
Secção 18.1

S.P. Dandamudi, *Fundamentals of Computer Organization and Design*, Ed. Springer, 2003

Espaço de endereçamento de um programa

- Para ser executado, um programa tem de ser trazido para memória central – carregamento de um *ficheiro executável*.
- O endereço de carregamento de um programa não é conhecido na altura da sua compilação e ligação
- Um ficheiro executável pode ser carregado em diferentes endereços físicos

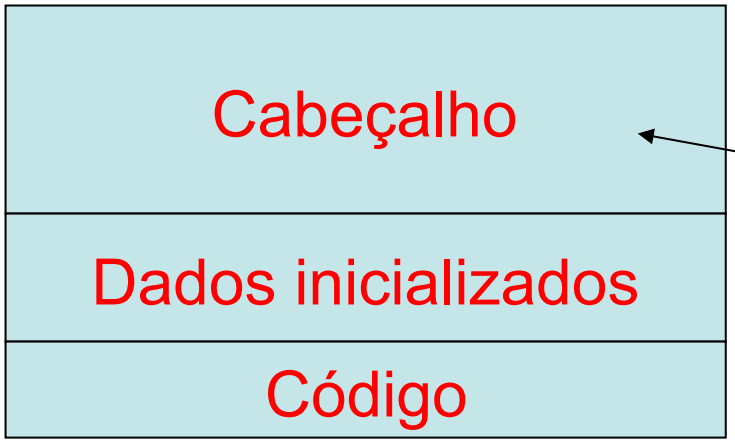
Obtenção do ficheiro executável



O que faz o compilador?

- Produz um ficheiro objecto
  - ✓ Como não se sabe com que outros módulos vai ser ligado assume que os endereços do módulo se iniciam em 0
- Traduz as instruções da linguagem de alto nível para código máquina do CPU
  - ✓ Aos símbolos definidos no módulo é atribuído temporariamente um endereço que é a distância em relação ao início do módulo
  - ✓ Aos símbolos que não se encontram no módulo são inseridos numa tabela de símbolos a resolver

Ficheiro objecto



Tamanho dos dados inicializados, tamanho da pilha, símbolos externos a importar, símbolos públicos a exportar

O que faz o ligador?

- Funde ficheiros objecto
  - ✓ funde múltiplos ficheiros objecto *recolocáveis* (.o) num único ficheiro objecto *executável* que pode ser carregado para memória pelo carregador.
- Resolve referências externas
  - ✓Como parte do processo de fusão, resolve *referências externas*: referência a um símbolo definido noutro ficheiro objecto.
- Recoloca símbolos dentro de cada ficheiro objecto
  - ✓Soma ao deslocamento associado a cada símbolo definido dentro do módulo um deslocamento constante correspondente ao endereço inicial no ficheiro objecto.

O que faz o ligador (cont.)?

- Recolocação de símbolos
  - ✓Recoloca *símbolos* da sua posição relativa nos ficheiros .o para novas posições absolutas no executável.
  - ✓Actualiza todas as referências a estes símbolos para reflectir as suas novas posições.
    - ✓As referências podem ser a código ou dados
      - ✓**código:** `a(); /* ref ao símbolo a */`
      - ✓**dados:** `int *xp=&x; /* ref ao símbolo x */`
    - ✓Por causa desta operação de modificação, o ligador é, às vezes, chamado editor de ligação (*link editor)*.

Porque é que há ligadores?

- Modularidade
  - ✓O programa pode ser escrito como um conjunto de pequenos ficheiros fonte, em vez de um monolítico.
  - ✓Podem-se construir bibliotecas de funções comuns : biblioteca numérica, biblioteca *standard* do C
  - ✓Tempo:
    - ✓Mudar um ficheiro fonte, compilar, e voltar a ligar; não há necessidade de recompilar outros ficheiros fonte
  - ✓Espaço:
    - ✓ As bibliotecas de funções de uso geral podem ser agregadas num ficheiro único …
    - ✓ … mas os ficheiros executáveis só contêm o código para as funções realmente usadas.

Exemplo de um programa em C

m.c

```
int e=7;

int main() {
  int r = a();
  exit(0);
}
```

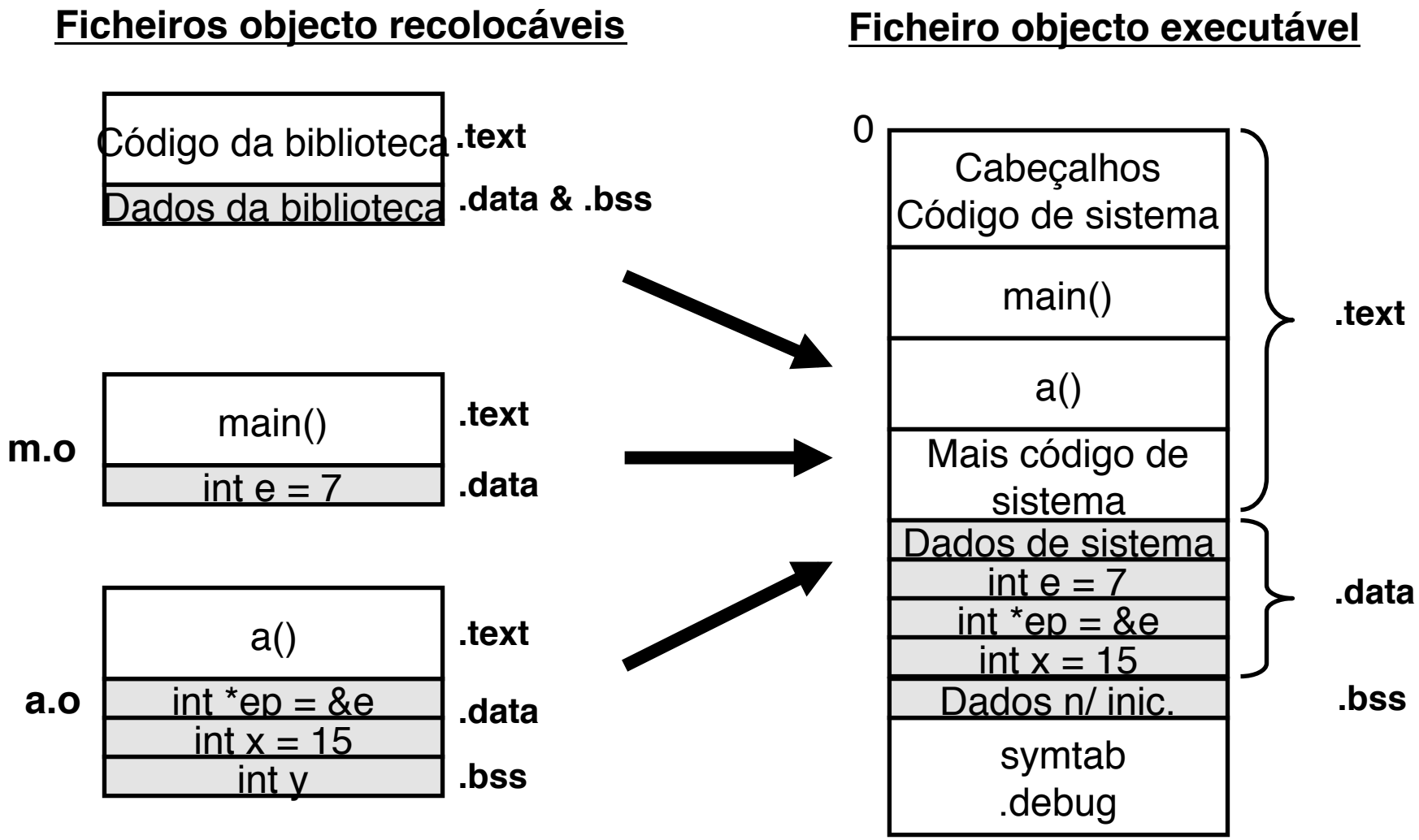
a.c

```
extern int e;

int *ep=&e;
int x=15;
int y;

int a() {
  return *ep+x+y;
}
```

Fusão dos fichs .o num executável



Espaço de endereçamento de um programa

- Para ser executado, um programa tem de ser trazido para memória central – carregamento do *ficheiro executável*

- Esta imagem em memória é a "Imagem do Processo":
  - ✓ é constituída por um conjunto de endereços contíguos
  - ✓ o conteúdo da memória inclui o código, dados e pilha do processo

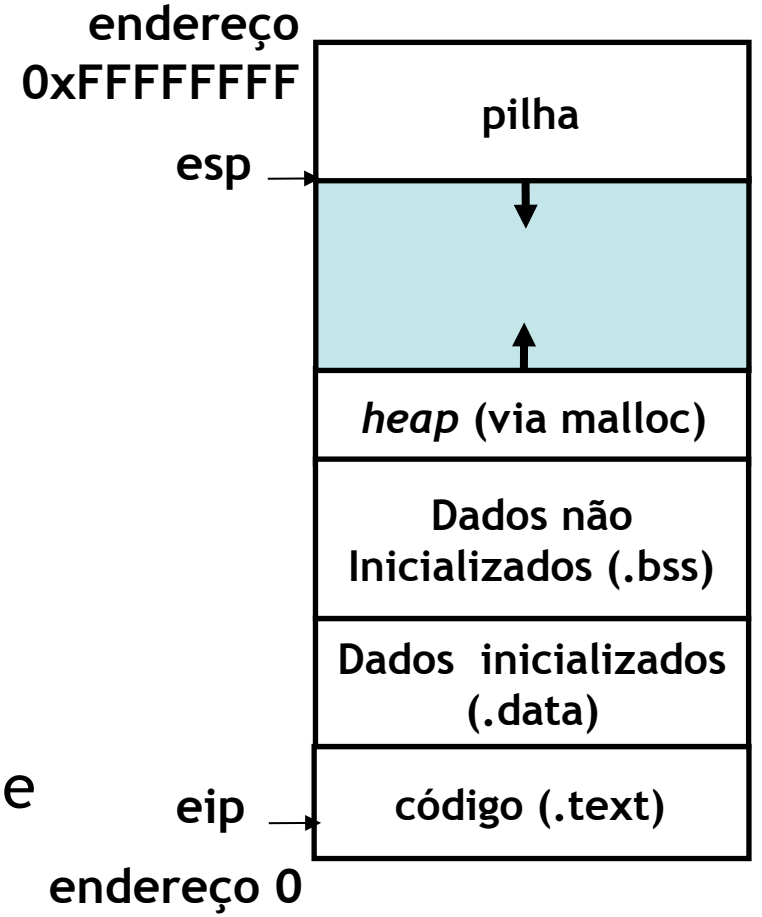
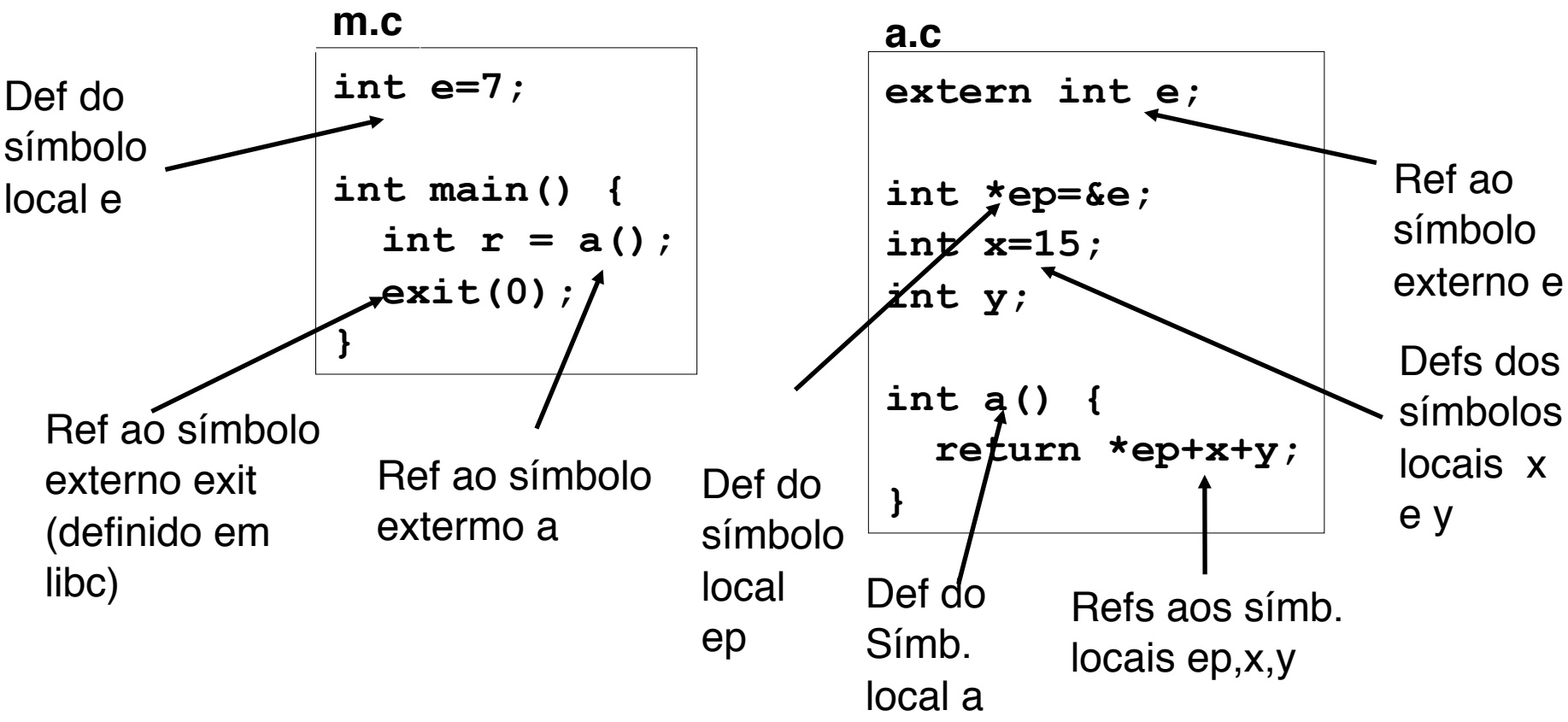


Imagem de memória de um processo Linux/x86 (simplificado)

Recolocação de símbolos e resolução de referências externas

Os símbolos são entidades lexicais que designam funções e variávies. Cada símbolo tem um *valor* (um endereço de memória).



Atribuição (binding) de endereços físicos às instruções e dados

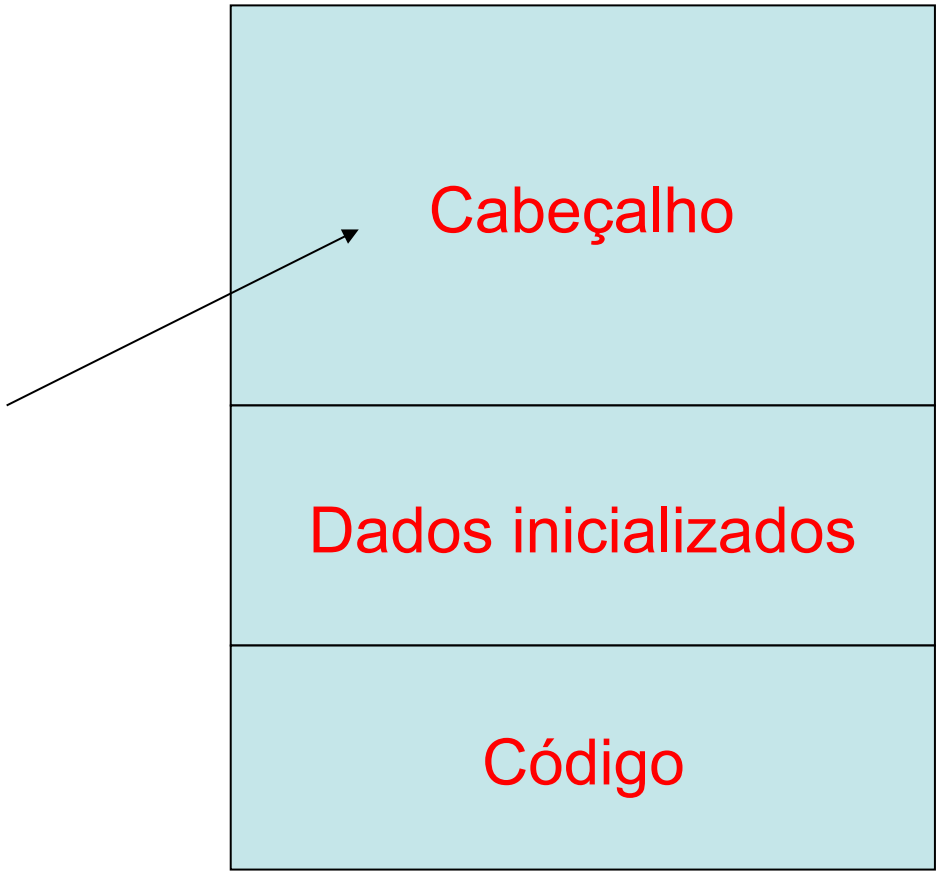
- **Compilação separada**: Não é possível atribuir endereços físicos às variáveis, estruturas de dados, linhas de programa, procedimentos. Não se sabe com que outros módulos este vai ser ligado, há referências a símbolos externos, por exemplo aos definidos em bibliotecas da linguagem.
- **Ligação**: É necessário gerar *código recolocável se o endereço de carregamento* não é conhecido na altura da compilação.
- **Carregamento**: Só nesta altura é que é pedido ao sistema de operação espaço na memória central para carregor o programa. Neste altura sabe-se o *endereço de carregamento*.

O ficheiro executável

Dimensões das várias zonas
- •Código (.text)
- •Dados inicializados (.data)
- •Dados não inicializados (.bss)
- •Pilha (.stack)

Valores iniciais para
- •Program counter (ponto de entrada do programa
- •Stack pointer (topo da zona reservada para a pilha)

Tabela de símbolos
- •Para “debug”



O que preciso para colocar o programa em execução? (1)

- Pedir ao Sistema de Operação para criar um *processo* para executar o programa
- Noção de **processo**:
  - ✓ processo é uma entidade activa
  - ✓ Executa um programa (ficheiro executável) que foi carregado na memória central.
  - Executa uma computação, a que corresponde um estado que vai sendo mudado à medida que o processo vai progredindo na computação.
  - Do estado fazem parte
    - ✓ Os registos do CPU que executa o programa
    - ✓ As zonas de memória do programa (código, dados e pilha)
    - ✓ Os dispositivos de entrada/saída que estão a ser usados

O que preciso para colocar o programa em execução? (2)

- **Inicialização do estado do processo:**
  - ✓ Executado pelo carregador e pelo SO a partir de informação que está no ficheiro executável
- Inicialização dos vários componentes do estado
  - Registos do CPU
    - ✓ PC e ESP inicializados a partir do cabeçalho do ficheiro executável
  - Memória central (imagem do processo)
    - ✓ Código e dados inicializados com conteúdos obtidos no ficheiro executável
    - ✓ Dados não inicializados e pilha – espaço reservado a partir de indicações no cabeçalho do ficheiro executável
  - Canais de entrada/saída:
    - ✓ O processo pode usar os canais 0 (entrada- teclado), 1 (saída – écrã) e 2 (erros – écrã)

Imagem do processo (1)

- Conteúdo das zonas de memória que pertencem ao processo
- Constituído por um conjunto de endereços contíguos de L bytes de comprimento
- Pode-se endereçar qualquer byte entre 0 e L-1

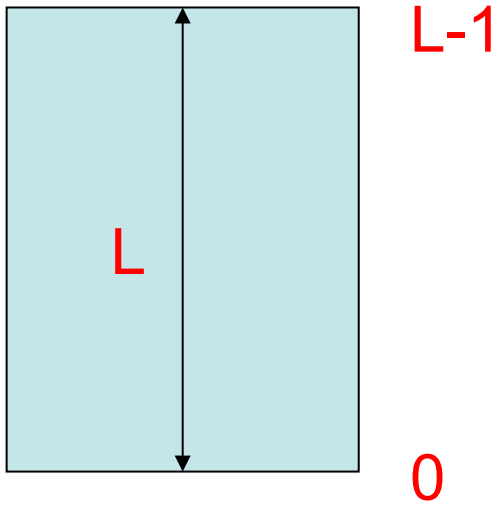
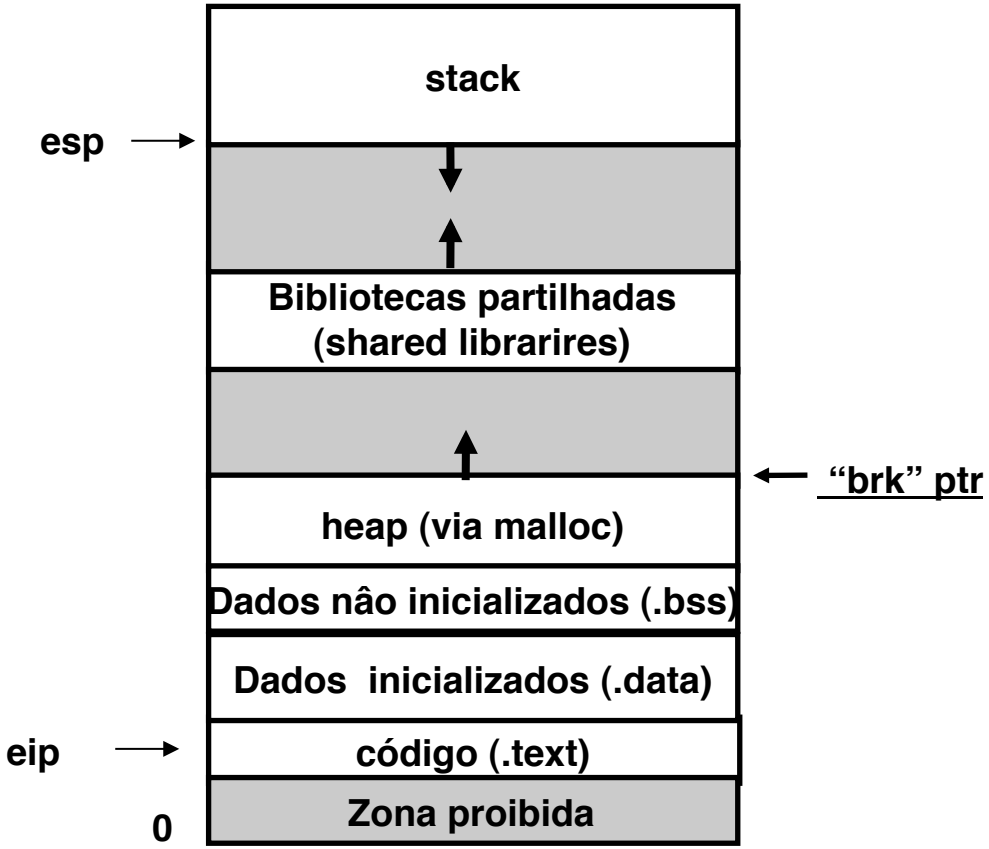


Imagem do processo (2)

Imagem de memória de um processo Linux/x86



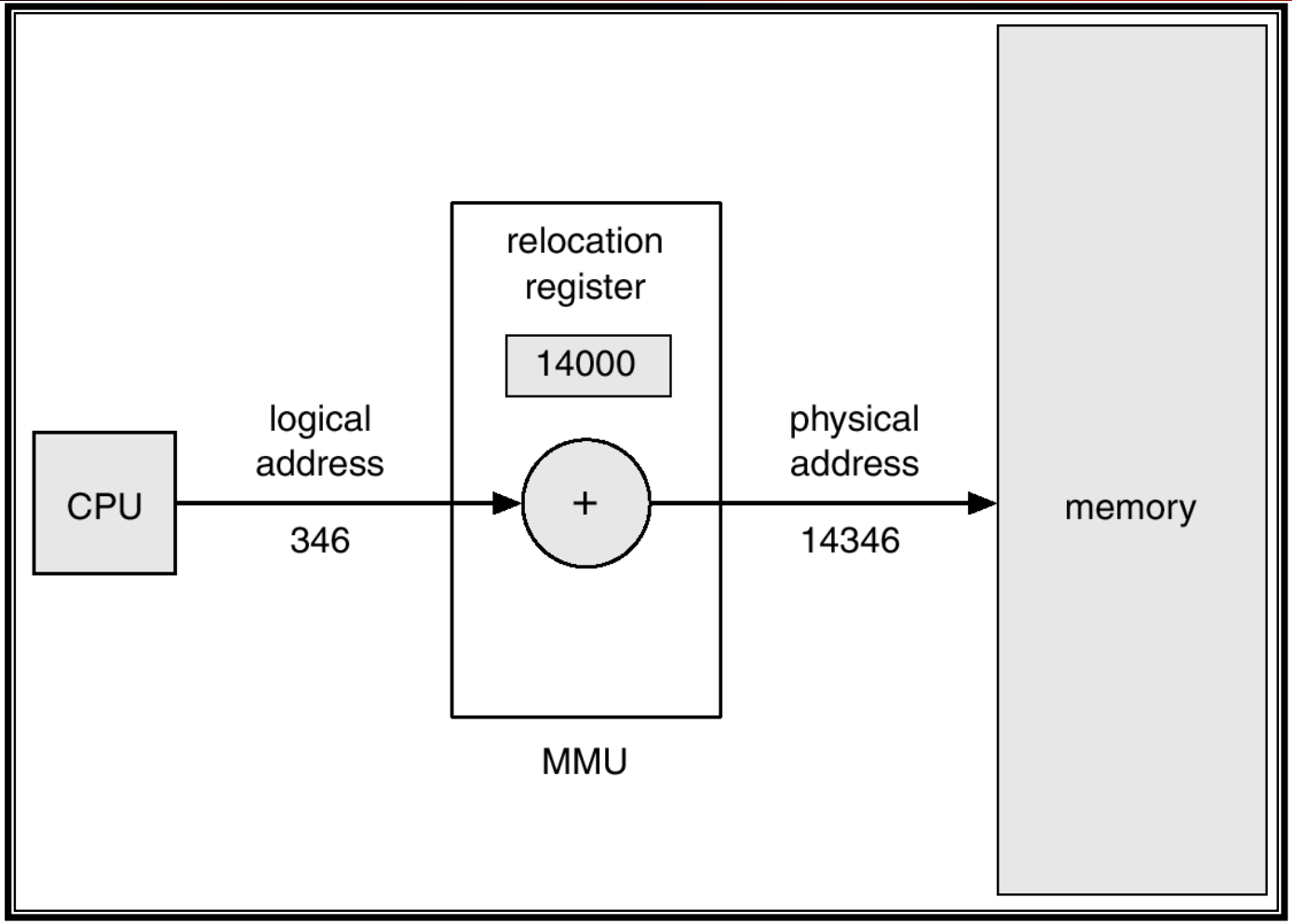
Espaços de endereços lógicos (ou virtuais) e físicos

- Em cada momento, estão carregados em memória imagens (código + dados + pilha)de vários processos
- Cada processo tem um espaço de endereçamento lógico que é separado do dos outros processos
- O conceito de um *espaço de endereçamento lógico* é independente do *espaço de endereços físico*
  - ✓ *Endereços lógicos* – gerados pelo CPU; também conhecidos por *endereços virtuais*.
  - ✓ *Endereços físicos* – endereços recebidos pela memória física
- os endereços virtuais e físicos diferem.

Unidade de gestão de memória (MMU – memory management unit)

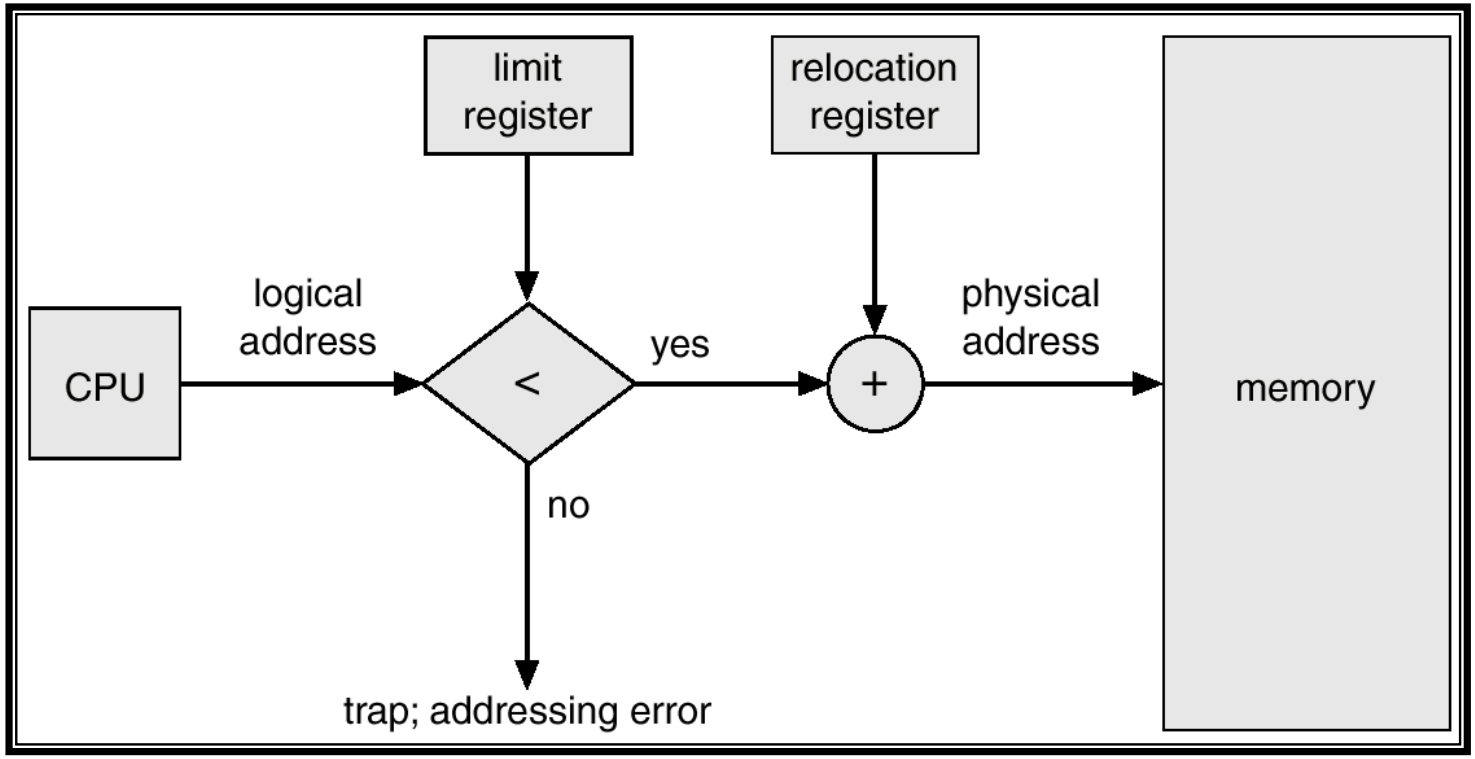
- Dispositivo hardware que faz a correspondência (mapping) entre endereços virtuais e físicos.
- O programa gera endereços *virtuais*; não controla os endereços físicos.
- As MMUs mais simples usam um *registo de recolocação*: o valor deste registo é somado a todos os endereços gerados pelo programa antes de estes serem enviados para a memória.

MMU com registo de recolocação



Esta técnica permite carregar o programa em qualquer posição da memória física

MMU com registo de recolocação (registo base) e registo limite



Este hardware permite não só a recolocação da imagem dos processos, como impede que um processo tenha acesso às imagens de outros processos