ISA – CHAMADAS A FUNÇÕES

TeSP de Aplicações Móveis

André Martins Pereira

IPCA
INSTITUTO POLITÉCNICO
DO CÁVADO E DO AVE
ESCOLA SUPERIOR DE TECNOLOGIA

ETAPAS

1. Invocação da função na função invocadora
2. Execução da função
3. Retorno à função invocadora

UM EXEMPLO

```
int addP(int a, int b){
    return a+b;
}
```

```
addP:
    pushl %ebp                  #guardar %ebp
    movl %esp,%ebp              #atualizar %esp

    movl 8(%ebp),%eax           #%eax = a
    movl 12(%ebp),%ebx          #%ebx = b
    addl %ebx,%eax              #a+b

    leave                       #recuperar %esp e %ebp
    ret                         #retornar
```

Arranque

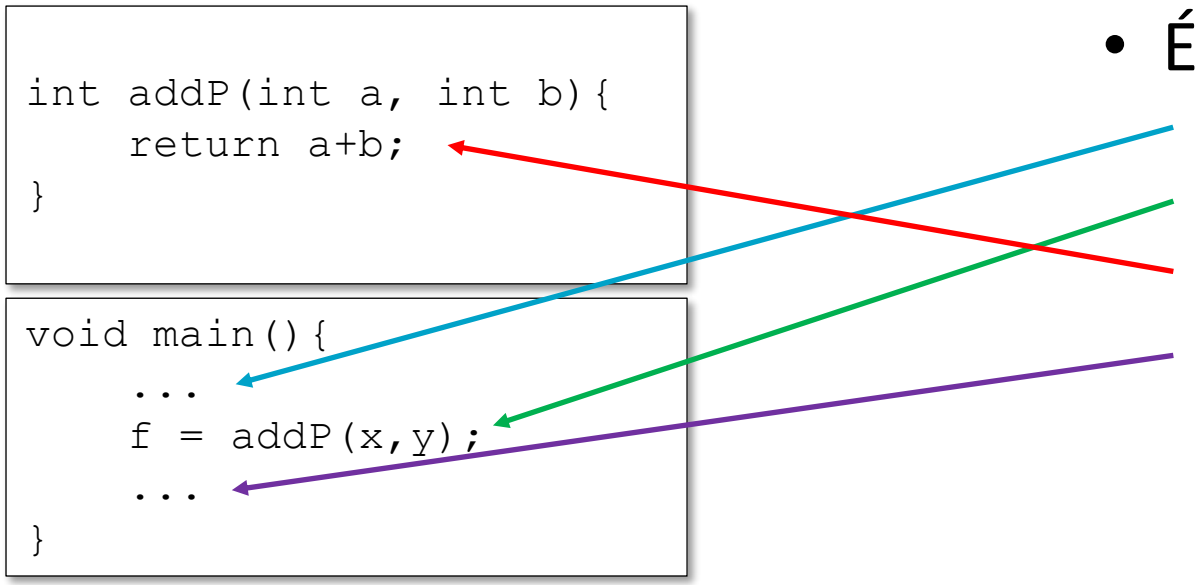
Corpo

Término

UM EXEMPLO

```
int addP(int a, int b){
    return a+b;
}
```

```
void main(){
    ...
    f = addP(x,y);
    ...
}
```

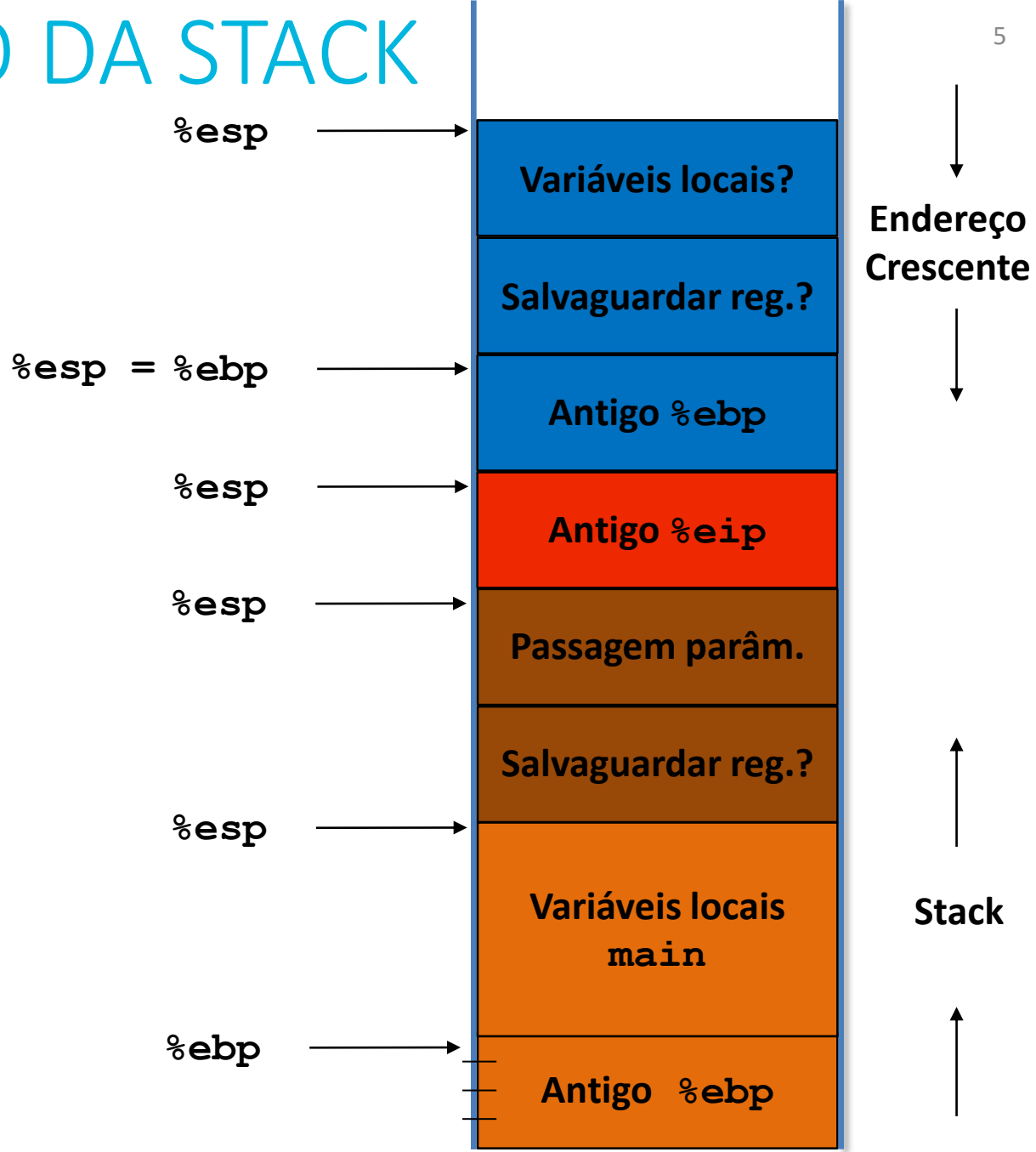


- É necessário perceber o que acontece:
  - Antes de chamar `addP`
  - Aquando da chamada a `addP`
  - Durante a execução de `addP`
  - No regresso à função `main`

`main` `addP`

CONSTRUÇÃO DA STACK

1. Antes de invocar addP
2. Preparar para invocar addP
   - Salvaguardar registos?
   - Passagem de parâmetros
3. Invocar addP ...
   - ... e guardar endereço de retorno
     1. Início de addP
        - Atualizar base pointer
        - Salvaguardar registos?
     2. Corpo de addP
     3. Término de addP
        - Recuperar registos?
        - Recuperar antigo base pointer
        - Regressar à função invocadora (main)
4. Terminar invocação de addP
   - Libertar espaço com parâmetros na stack
   - Recuperar registos?



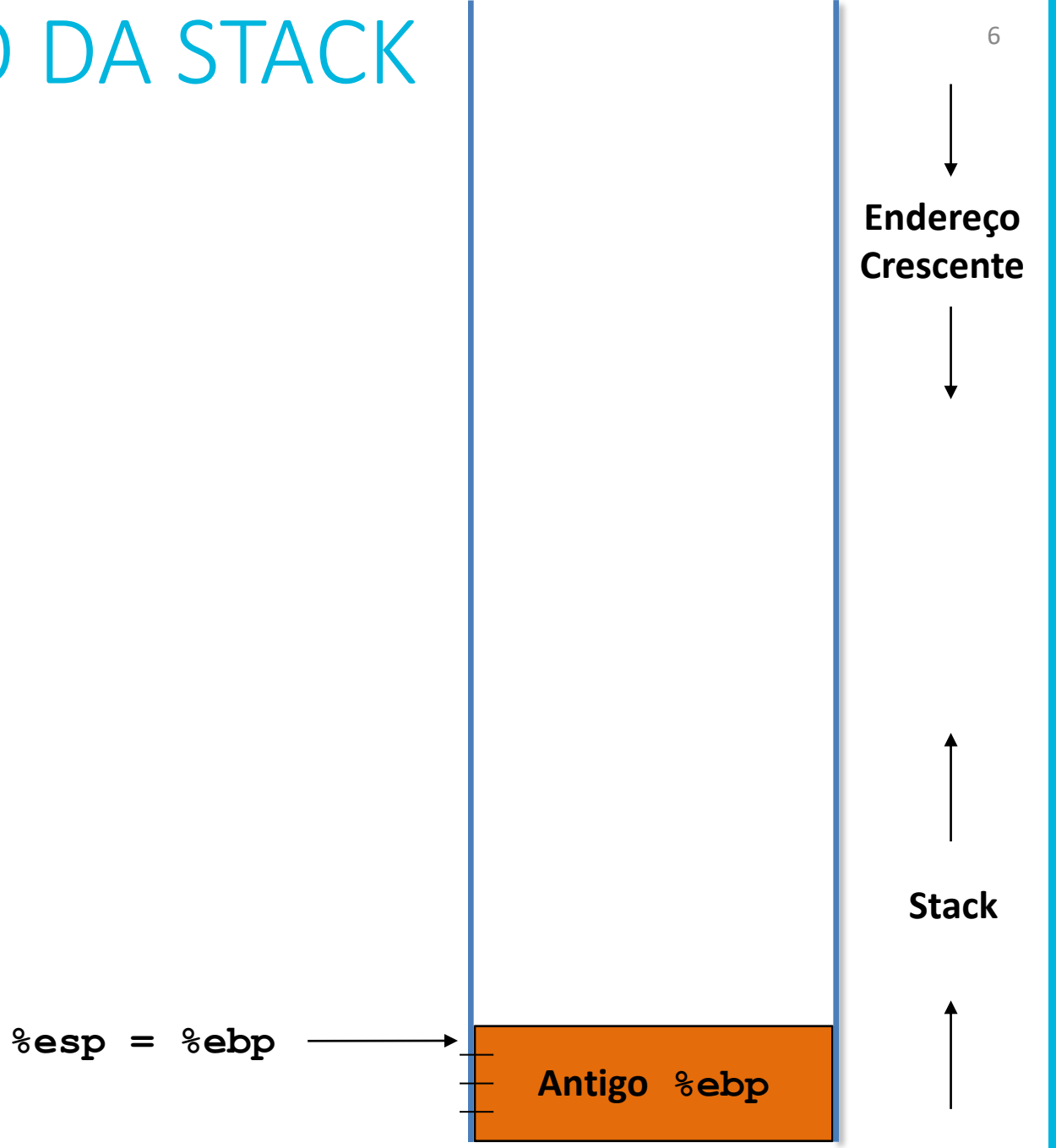
CONSTRUÇÃO DA STACK

`main`

1. **Antes de invocar addP**

```
int addP(int a, int b){
    return a+b;
}
```

```
void main(){
    (...)
    f = addP(x,y);
    (...)
}
```



CONSTRUÇÃO DA STACK

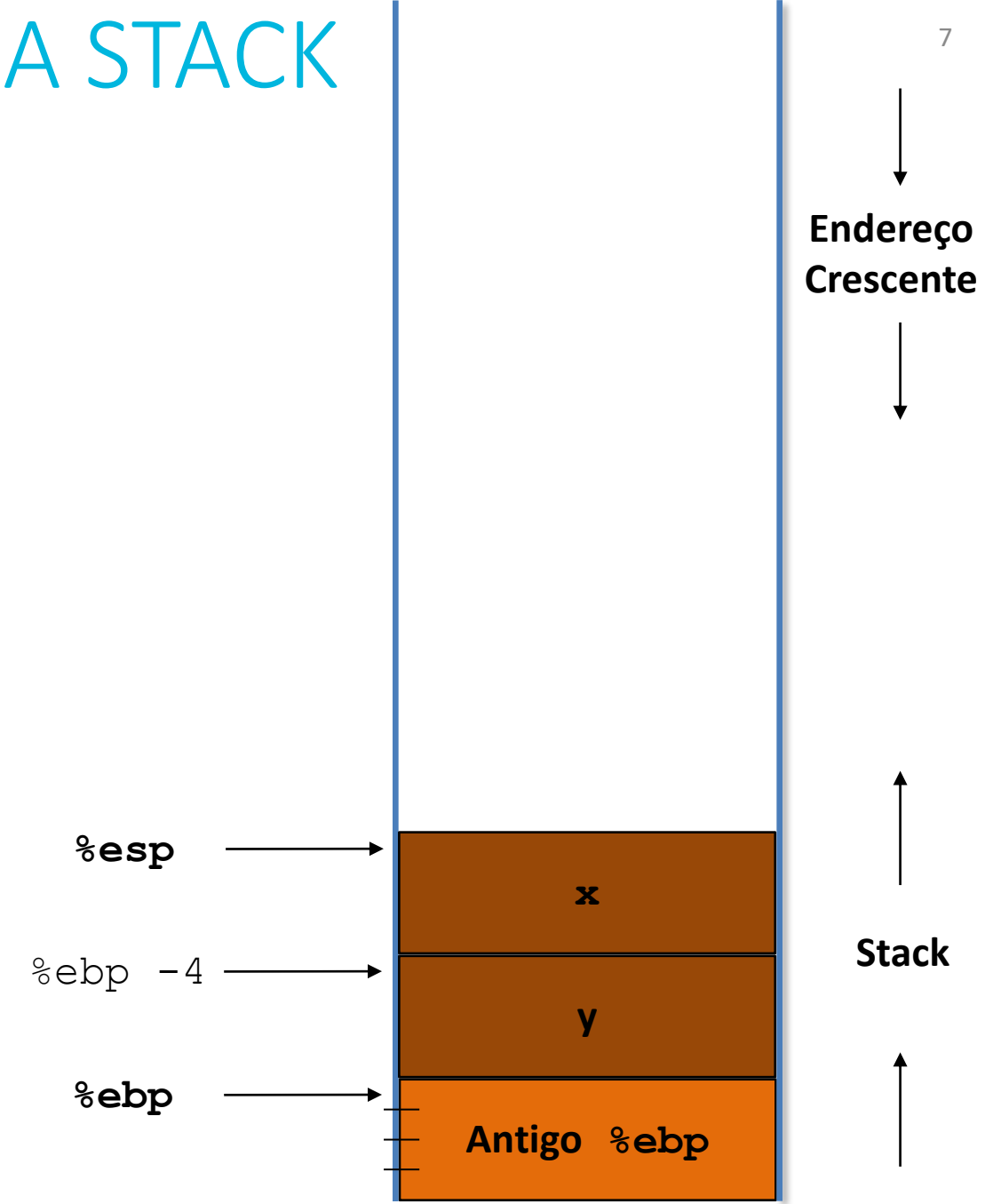
`main`

2. Preparar para invocar addP

- **Salvaguardar registos?** Não…
- **Passagem de parâmetros**

```
int addP(int a, int b){
    return a+b;
}
```

```
void main(){
    (...)
    f = addP(x,y);
    (...)
}
```

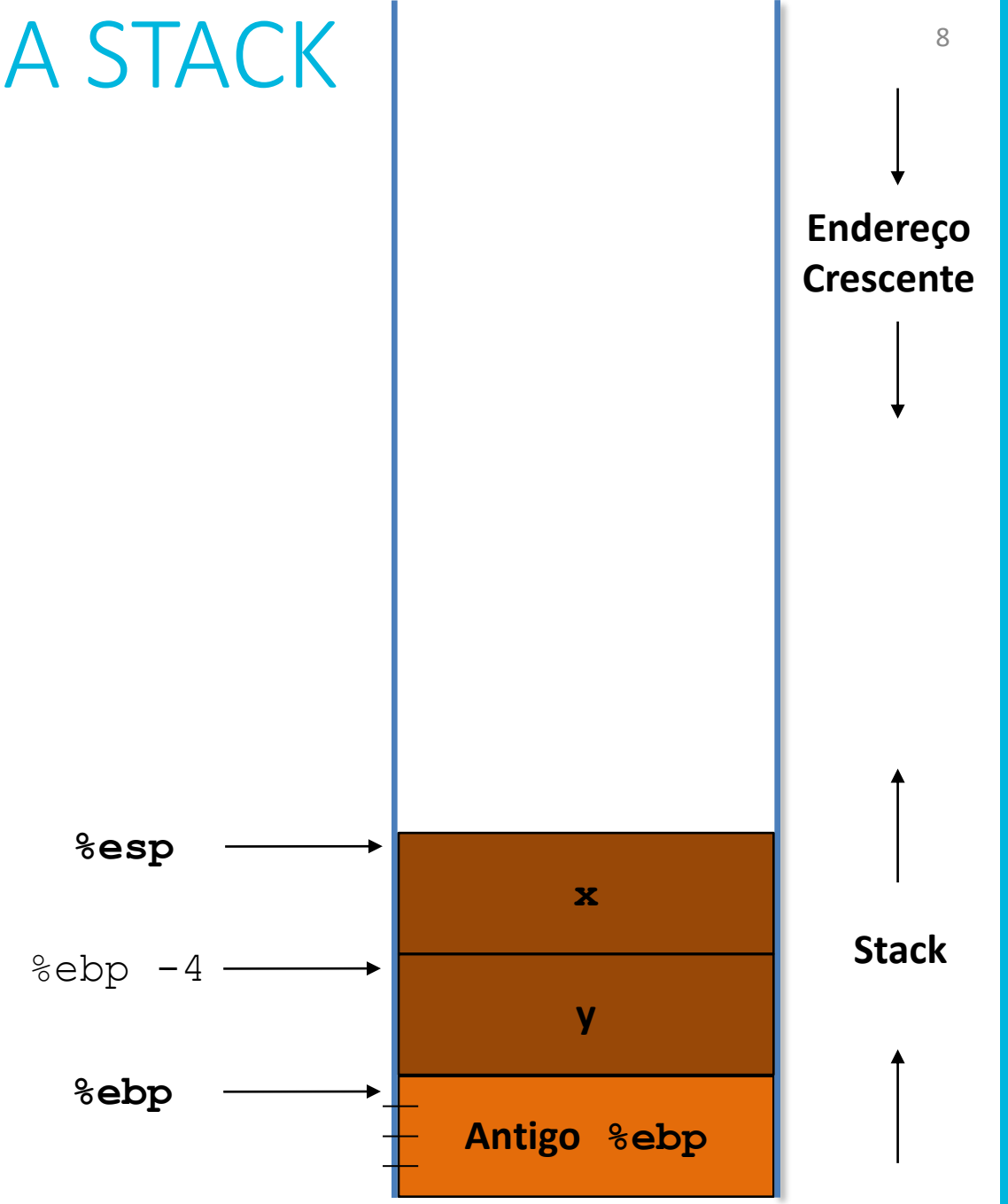


CONSTRUÇÃO DA STACK

`main`

2. **Preparar para invocar addP**
   - **Salvaguardar registos?** Não…
   - **Passagem de parâmetros**

```
pushl %ebx
pushl %eax
```



CONSTRUÇÃO DA STACK

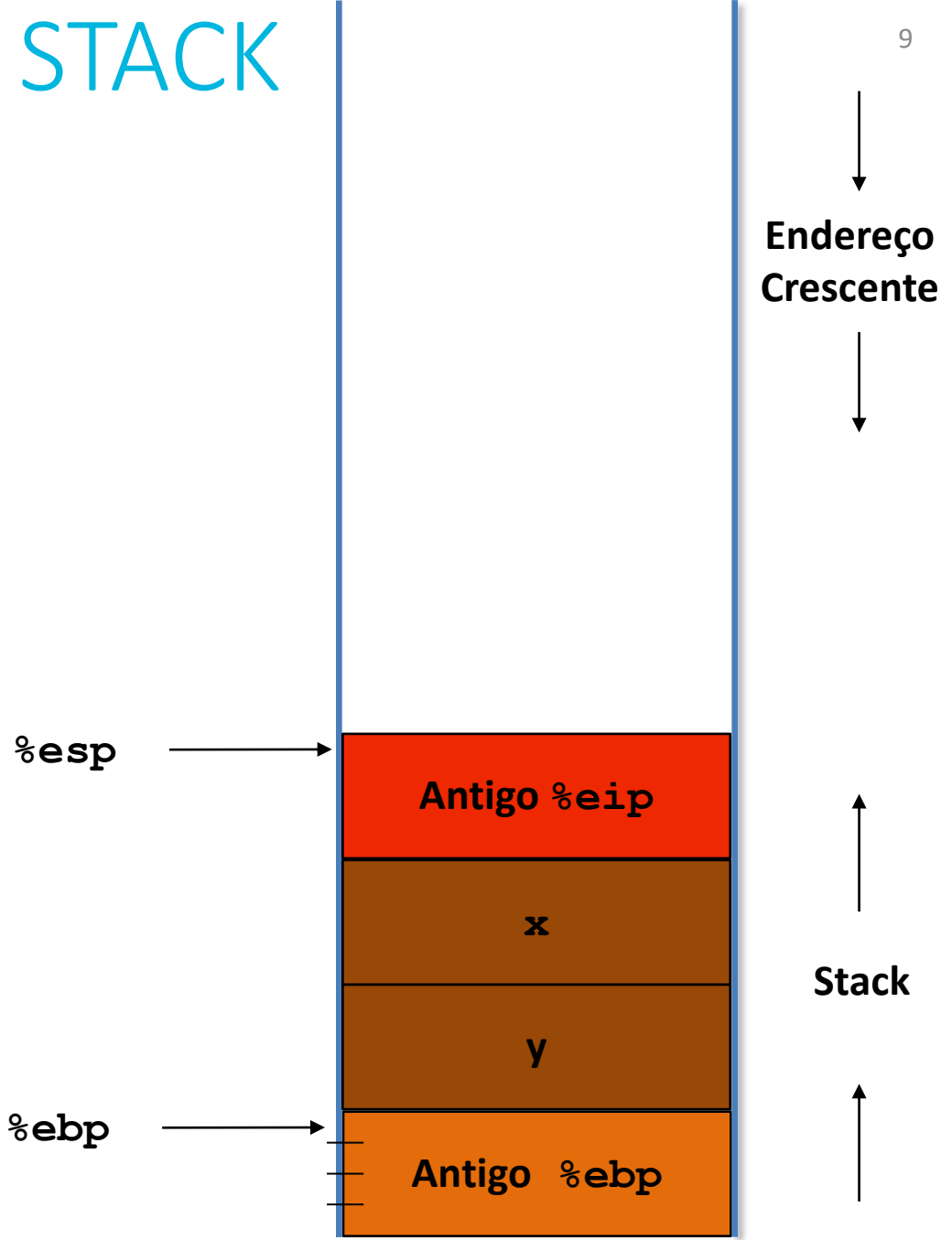
`main`

1. **Invocar addP ...**
   - **... e guardar endereço de retorno**

```
int addP(int a, int b){
    return a+b;
}
```

```
void main(){
    (...)
    f = addP(x,y);
    (...)
}
```

```
call addP
```



CONSTRUÇÃO DA STACK

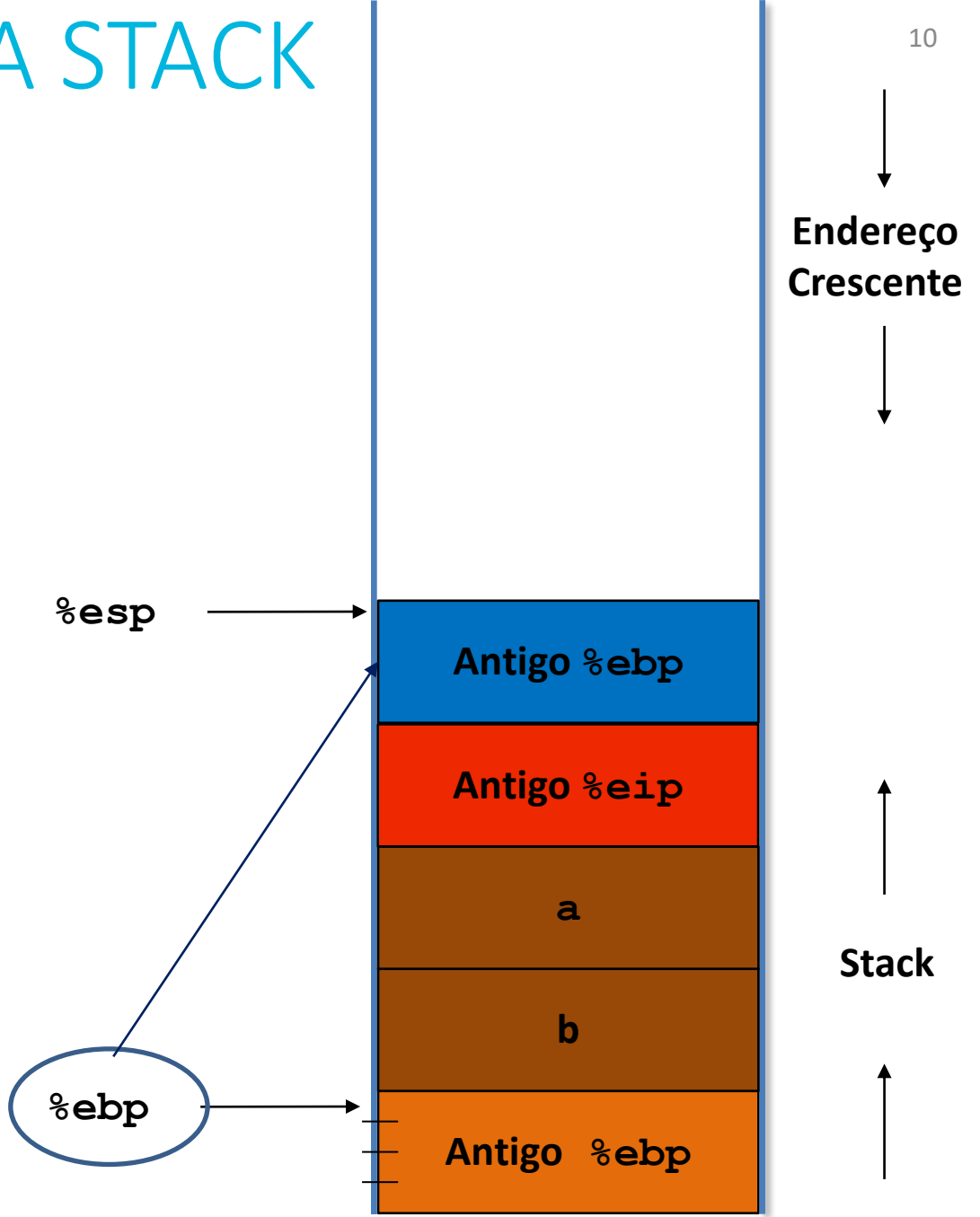
addP

1. **Início de addP**
   - **Atualizar %ebp**

```
int addP(int a, int b){
     return a+b;
}
```

```
void main(){
     (...)
     f = addP(x,y);
     (...)
}
```

```
swap:
     pushl %ebp
```



CONSTRUÇÃO DA STACK

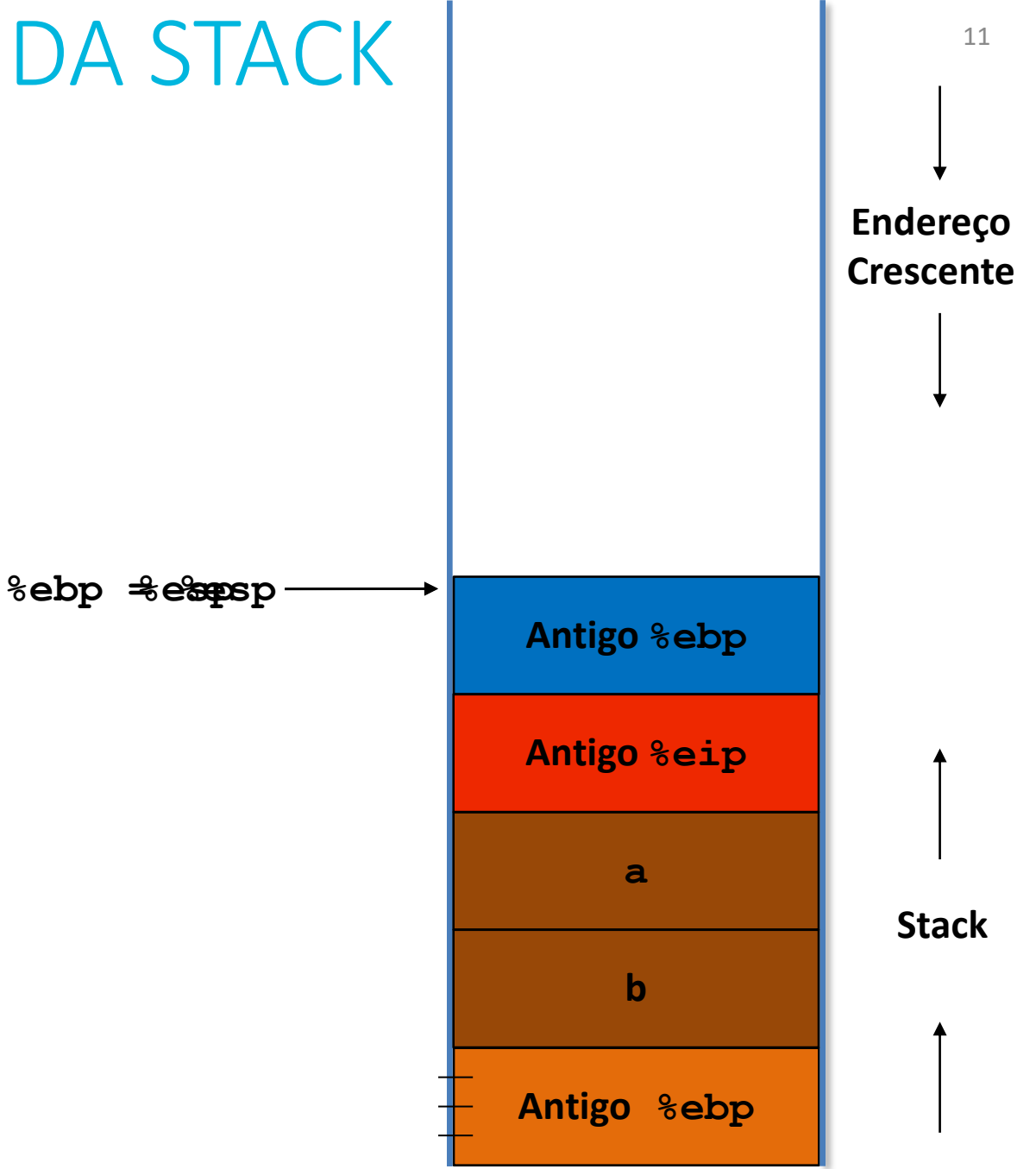
addP

1. **Início de addP**
   - **Atualizar `%ebp`**
   - **Salvaguardar reg.?** Não…
   - **Var. locais?** Não…

```
int addP(int a, int b){
    return a+b;
}
```

```
void main(){
    (...)
    f = addP(x,y);
    (...)
}
```

```
swap:
    pushl %ebp
    movl %esp, %ebp
```



addP

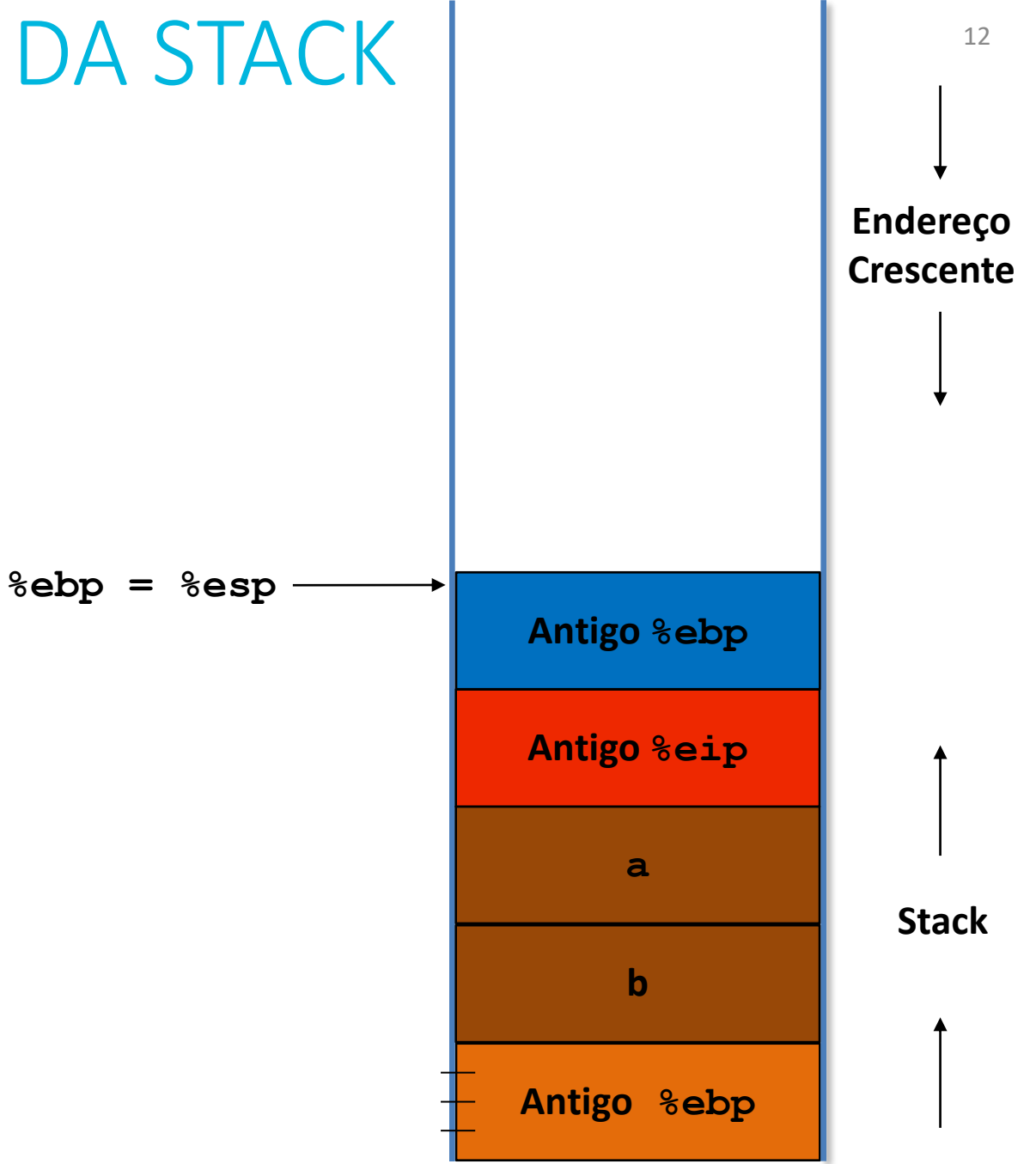
CONSTRUÇÃO DA STACK

2. Corpo de addP

```
int addP(int a, int b){
    return a+b;
}
```

```
void main(){
    (...)
    f = addP(x,y);
    (...)
}
```

```
    movl 8(%ebp),%eax
    movl 12(%ebp),%ebx
    addl %ebx,%eax
```



CONSTRUÇÃO DA STACK

addP

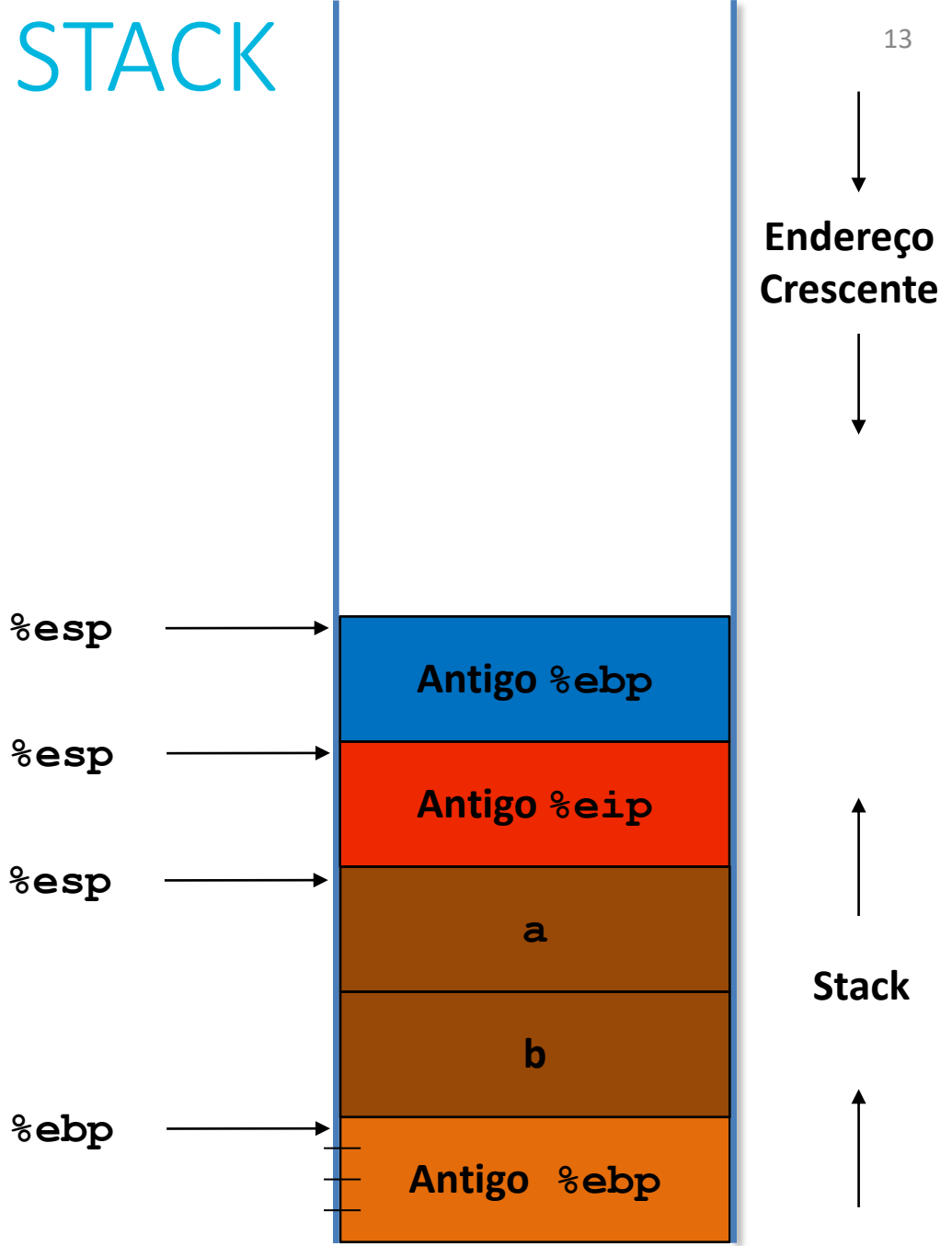
3. **Término de `addP`**
   - **Libertar espaço var. locais?** Não…
   - **Recuperar registos?** Não…
   - **Recuperar `%ebp` e `%esp`**
   - **Regressar à `main`**

```
int addP(int a, int b){
    return a+b;
}
```

```
void main(){
    (...)
    f = addP(x,y);
    (...)
}
```

```
    movl %ebp,%esp
    popl %ebp
         OU
    leave

    ret
```



addP

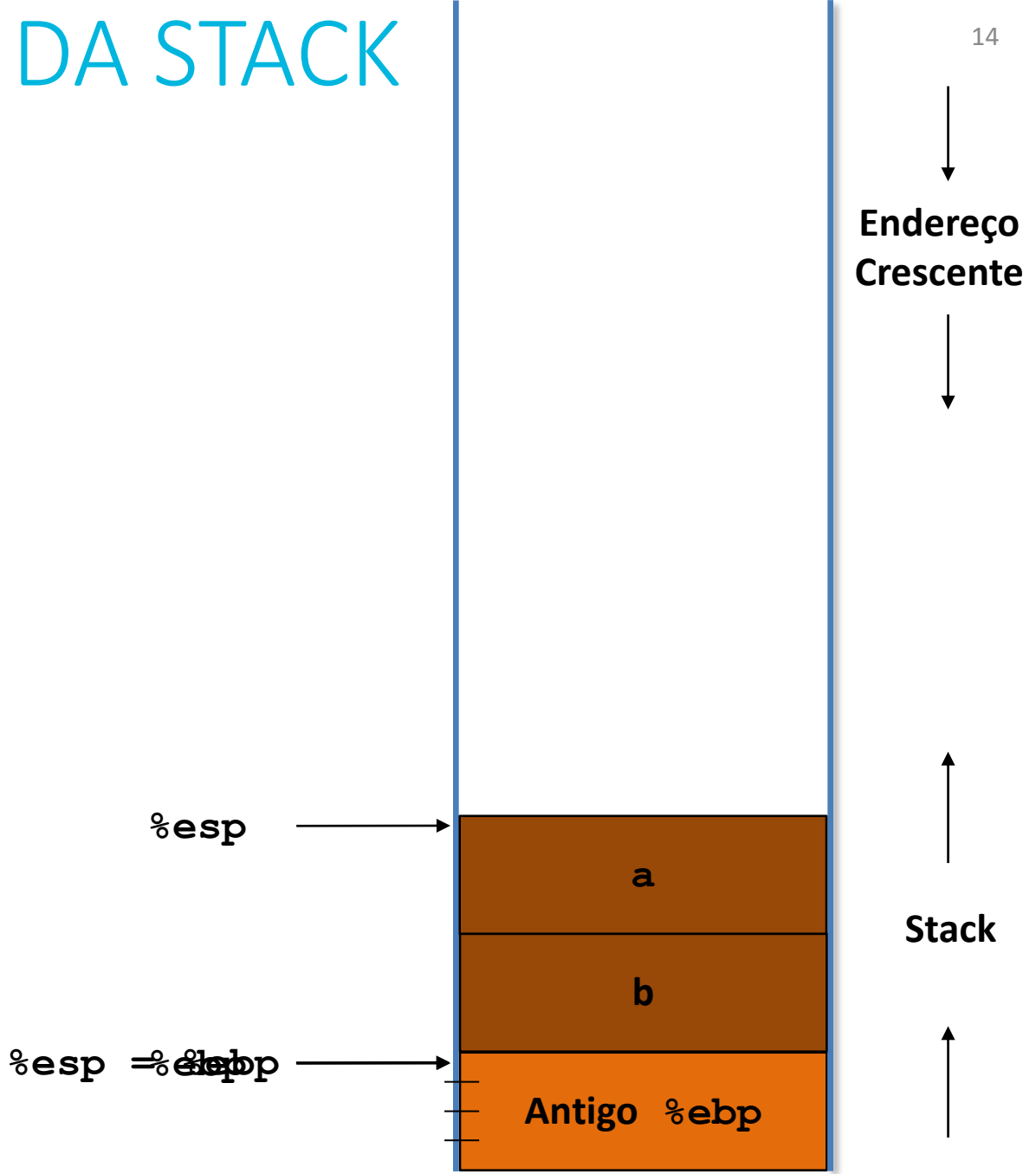
CONSTRUÇÃO DA STACK

1. **Regressar à função invocadora**
   - **Libertar espaço de parâmetros**
   - **Recuperar registos?** Não…

```
int addP(int a, int b){
    return a+b;
}
```

```
void main(){
    (...)
    f = addP(x,y);
    (...)
}
```

```
    addl $8, %esp
```



ISA – CHAMADAS A FUNÇÕES