Estrutura do tema ISA do IA-32

1. Desenvolvimento de programas no IA-32 em Linux
2. Acesso a operandos e operações
3. Suporte a estruturas de controlo
4. Suporte à invocação/regresso de funções
5. Análise comparativa: IA-32 *vs.* x86-64 e RISC (MIPS e ARM)
6. Acesso e manipulação de dados estruturados

Mecanismos necessários

(suporte do ISA a funções/procedimentos)

Passagem do controlo

- Executar o código da função
- Regressar ao ponto de chamada

Passagem de informação

- Argumentos da função
- Valor de retorno

Gestão da memória

- Reservar espaço na execução da função
- Libertar o espaço reservado no regresso

```
int main(...) {
  ...
  y = soma(x);
  print(y);
  ...
}
```

```
int soma(int i)
{
  int t = 3*i;
  int v[10];
  ...
  return v[t];
}
```

Mecanismos necessários

(suporte do ISA a funções/procedimentos)

Passagem do controlo

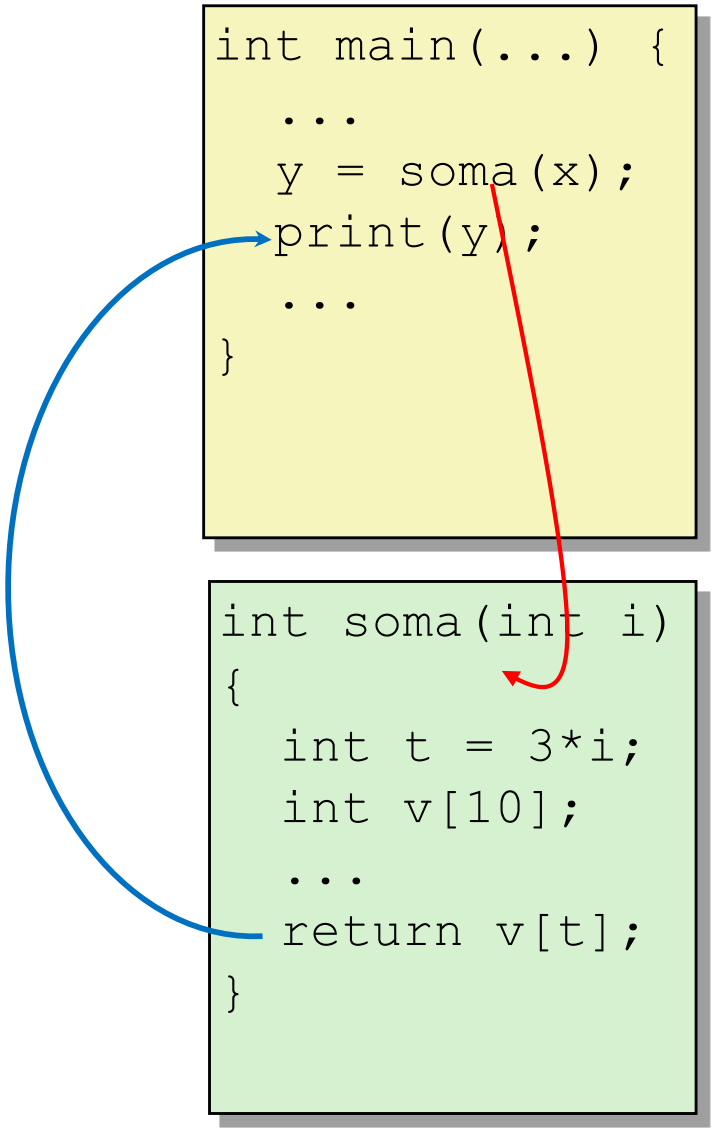
- Executar o código da função
- Regressar ao ponto de chamada

Passagem de informação

- Argumentos da função
- Valor de retorno

Gestão da memória

- Reservar espaço na execução da função
- Libertar o espaço reservado no regresso



Mecanismos necessários

(suporte do ISA a funções/procedimentos)

Passagem do controlo

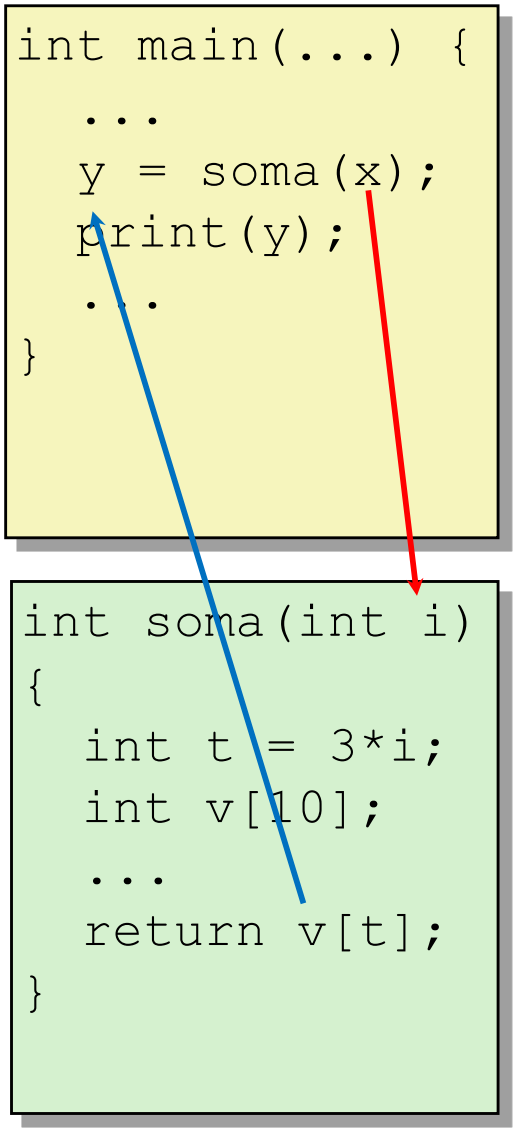
- Executar o código da função
- Regressar ao ponto de chamada

Passagem de informação

- Argumentos da função
- Valor de retorno

Gestão da memória

- Reservar espaço na execução da função
- Libertar o espaço reservado no regresso



Mecanismos necessários

(suporte do ISA a funções/procedimentos)

Passagem do controlo

- Executar o código da função
- Regressar ao ponto de chamada

Passagem de informação

- Argumentos da função
- Valor de retorno

Gestão da memória

- Reservar espaço na execução da função
- Libertar o espaço reservado no regresso

```
int main(...) {
  ...
  y = soma(x);
  print(y);
  ...
}
```

```
int soma(int i)
{
  int t = 3*i;
  int v[10];
  ...
  return v[t];
}
```

Suporte a funções: pilha (*stack*)

Passagem do controlo (IA-32)

`call func` – coloca na pilha o endereço de regresso e executa a função (i.é, *push IP+X; IP=end_de_func*)

`ret` – regressa da função usando o endereço armazenado na pilha (i.é., *pop IP*)

Passagem de informação (IA-32)

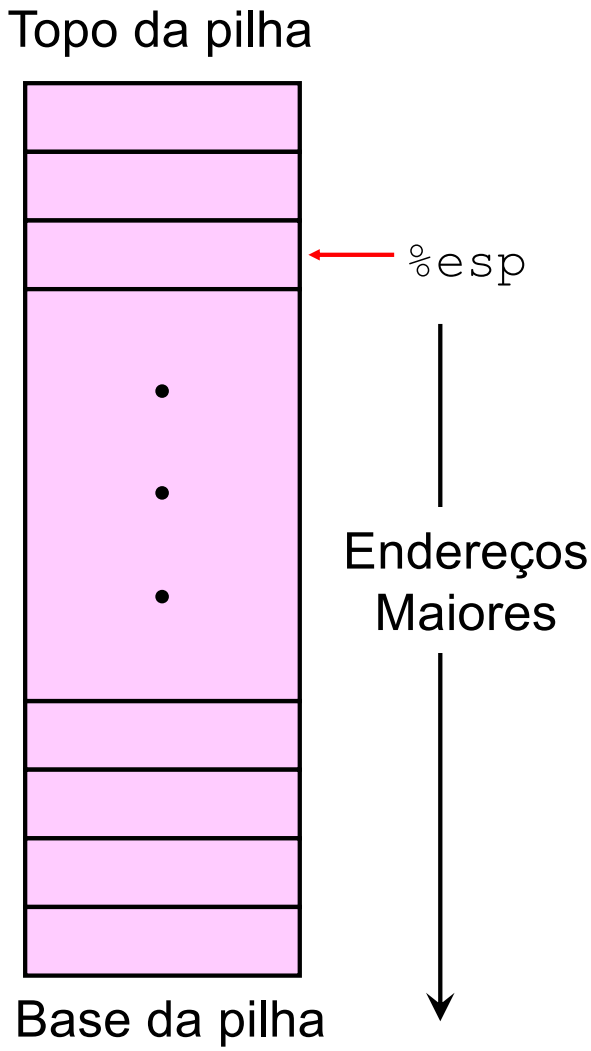
Os argumentos da função são colocados na pilha

Valor de retorno é colocado em `%eax`

Gestão da memória

`sub XX,%esp` - Reserva espaço para a função (por exemplo, variável local v[10])

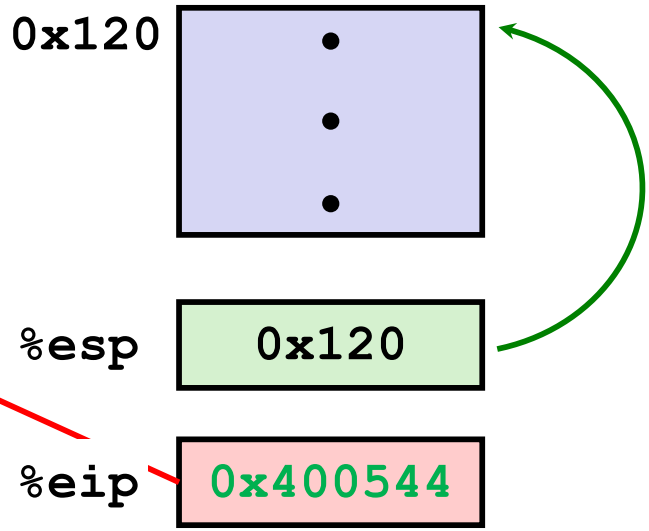
Liberta o espaço repondo o valor original do `%esp`



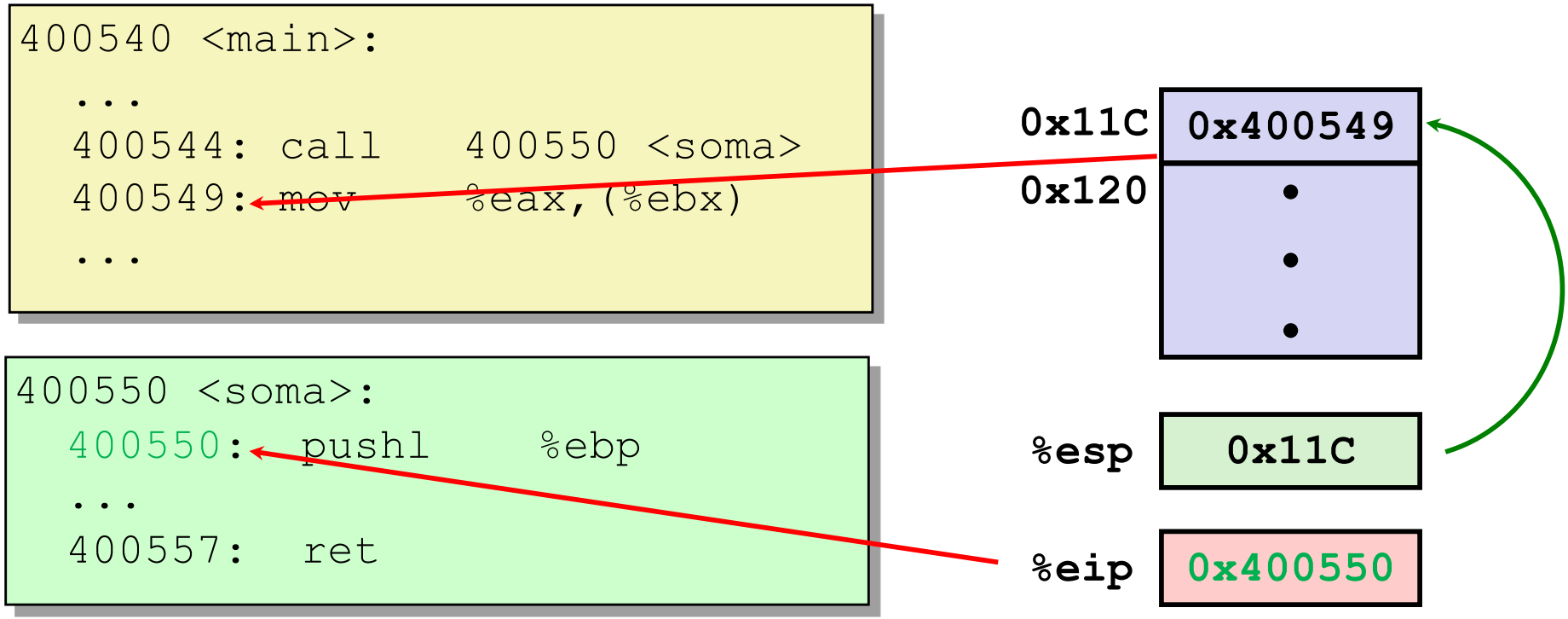
Passagem de controlo: exemplo (1)

```
400540 <main>:
  ...
  400544: call   400550 <soma>
  400549: mov    %eax,(%ebx)
  ...
```

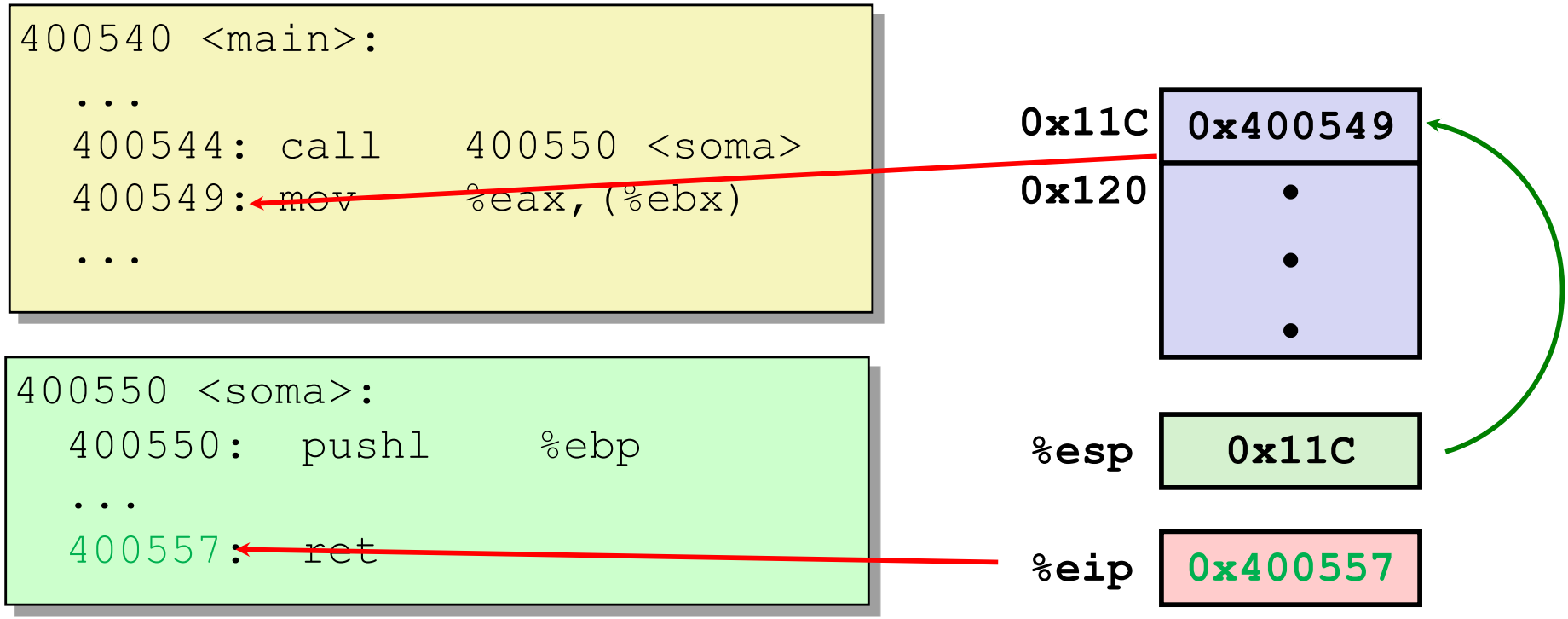
```
400550 <soma>:
  400550:  pushl    %ebp
  ...
  400557:  ret
```



Passagem de controlo: exemplo (2)



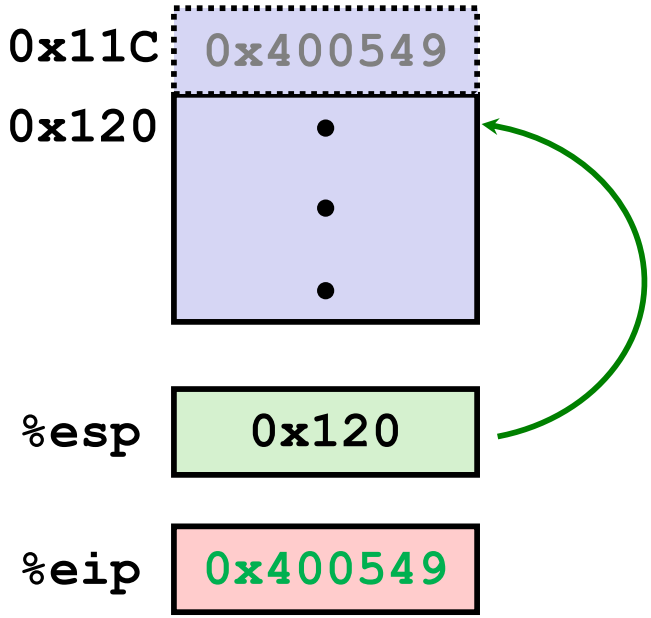
Passagem de controlo: exemplo (3)



Passagem de controlo: exemplo (4)

```
400540 <main>:
  ...
  400544: call   400550 <soma>
  400549: mov    %eax,(%ebx)
  ...
```

```
400550 <soma>:
  400550:  pushl    %ebp
  ...
  400557:  ret
```



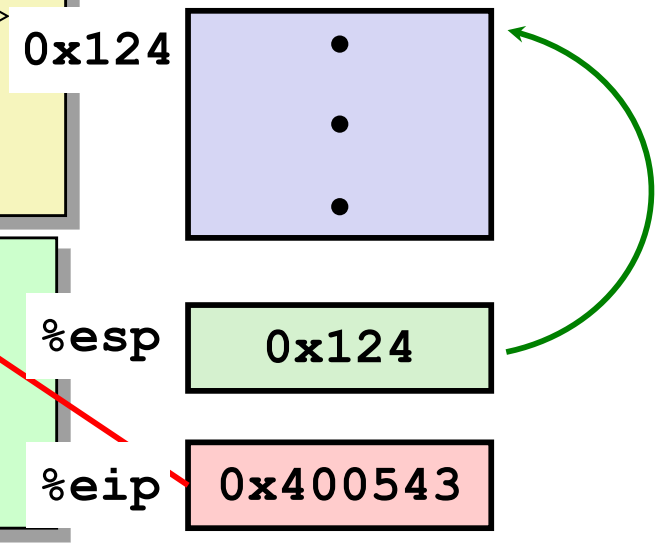
Passagem de informação: exemplo (1)

```
int main(...) {
   ...
   y = soma(x);
   print(y);
   ...
}
```

```
400540 <main>:
   ...
   400543: push %ebx
   400544: call 400550 <soma>
   400549: mov  %eax,(%ebx)
   ...
```

```
int soma(int i)
{
   int t = 3*i;
   ...
   return t;
}
```

```
400550 <soma>:
   400550:  mov 4(%esp),%eax
   ...
   400557:  ret
```



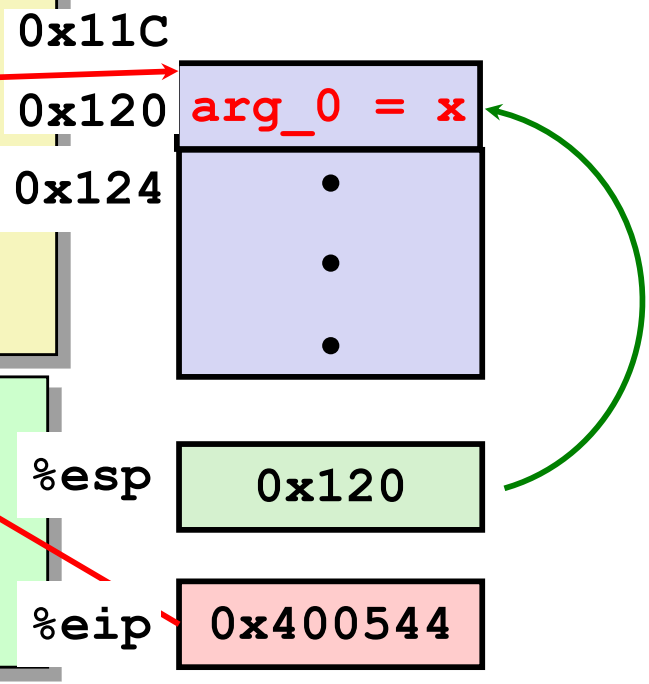
Passagem de informação: exemplo (2)

```
int main(...) {
  ...   // %ebx
  y = soma(x);
  print(y);
  ...
}
```

```
400540 <main>:
  ...
  400543: push %ebx
  400544: call 400550 <soma>
  400549: mov  %eax,(%ebx)
  ...
```

```
int soma(int i)
{
  int t = 3*i;
  ...
  return t;
}
```

```
400550 <soma>:
  400550:  mov 4(%esp),%eax
  ...
  400557:  ret
```



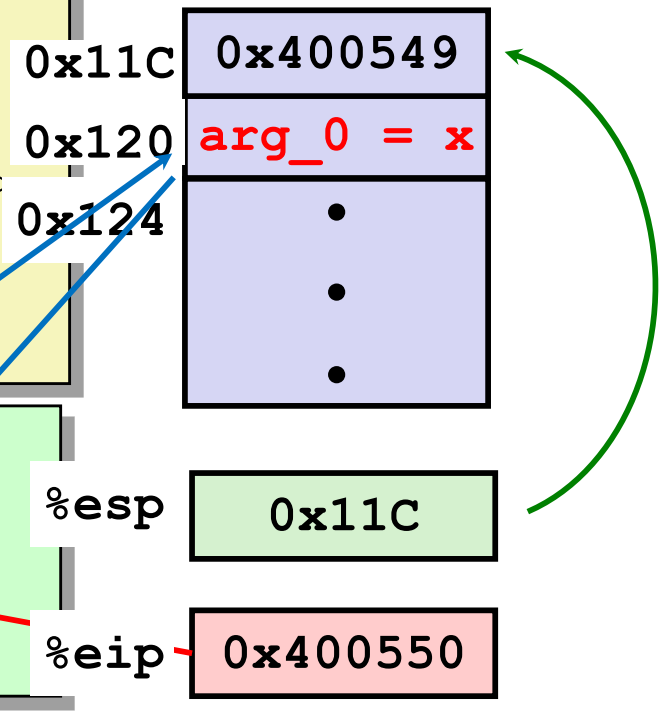
Passagem de informação: exemplo (3)

```
int main(...) {
  ...
  y = soma(x);
  print(y);
  ...
}
```

```
400540 <main>:
  ...
  400543: push %ebx
  400544: call 400550 <soma>
  400549: mov  %eax,(%ebx)
  ...
```

```
int soma(int i)
{
  int t = 3*i;
  ...
  return t;
}
```

```
400550 <soma>:
  400550:  mov 4(%esp),%eax
  ...
  400557:  ret
```



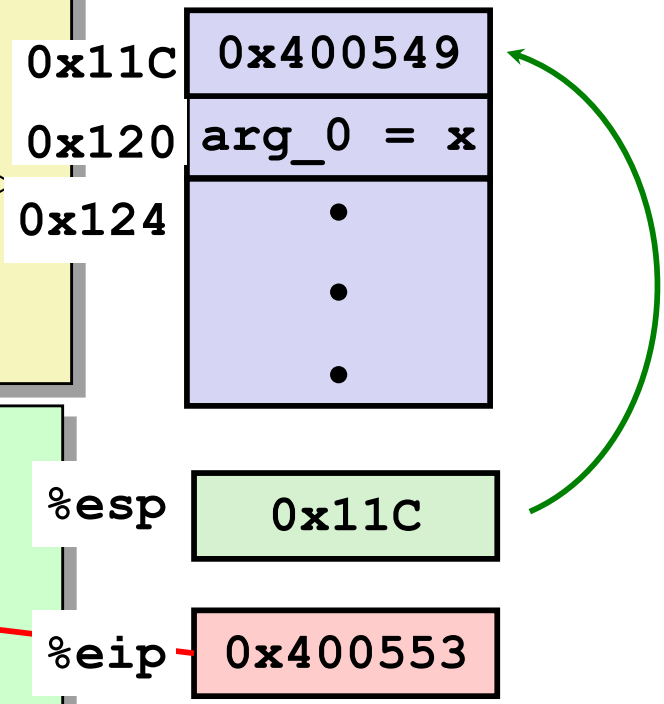
Passagem de informação: exemplo (4)

```
int main(...) {
  ...
  y = soma(x);
  print(y);
  ...
}
```

```
400540 <main>:
  ...
  400543: push %ebx
  400544: call 400550 <soma>
  400549: mov  %eax,(%ebx)
  ...
```

```
int soma(int i)
{
  int t = 3*i;
  ...
  return t;
}
```

```
400550 <soma>:
  400550:  mov 4(%esp),%eax
  400553:  imul $3,%eax
  ...
  400557:  ret
```



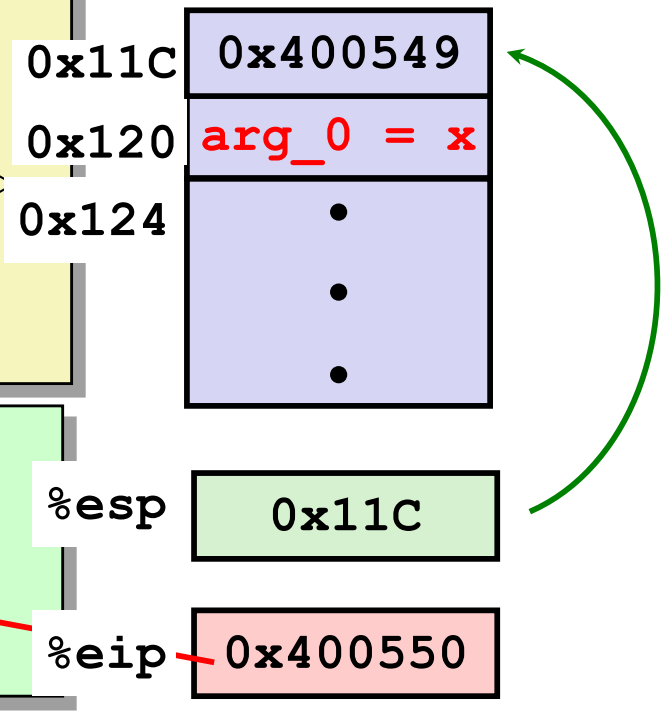
Gestão de memória: exemplo (1)

```
int main(...) {
   ...
   y = soma(x);
   print(y);
   ...
}
```

```
400540 <main>:
   ...
   400543: push %ebx
   400544: call 400550 <soma>
   400549: mov  %eax,(%ebx)
   ...
```

```
int soma(int i)
{
   int t = 3*i;
   int v[10];
   ...
   return t;
}
```

```
400550 <soma>:
   400550:  sub $40,%esp
   ...
   400557:  ret
```



Gestão de memória: exemplo (2)

```
int main(...) {
   ...
   y = soma(x);
   print(y);
   ...
}
```

```
400540 <main>:
   ...
   400543: push %ebx
   400544: call 400550 <soma>
   400549: mov  %eax,(%ebx)
   ...
```

```
int soma(int i)
{
   int t = 3*i;
   int v[10];
   ...
   return t;
}
```

```
400550 <soma>:
   400550:  sub $40,%esp
   ...
   400553:  add $40,%esp
   400557:  ret
```