Section 5: Procedures & Stacks

- Stacks in memory and stack operations
- The stack used to keep track of procedure calls
- Return addresses and return values
- Stack-based languages
- The Linux stack frame
- Passing arguments on the stack
- Allocating local variables on the stack
- Register-saving conventions
- Procedures and stacks on x64 architecture

Stack-Based Languages

- **Languages that support recursion**
  - e.g., C, Pascal, Java
  - Code must be *re-entrant*
    - Multiple simultaneous instantiations of single procedure
  - Need some place to store state of each instantiation
    - Arguments
    - Local variables
    - Return pointer
- **Stack discipline**
  - State for a given procedure needed for a limited time
    - Starting from when it is called to when it returns
  - Callee always returns before caller does
- **Stack allocated in *frames***
  - State for a single procedure instantiation

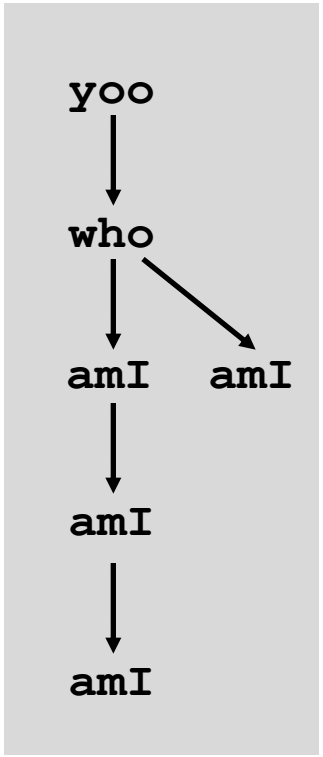
Call Chain Example

```
yoo(…)
{
    •
    •
    who();
    •
    •
}
```

```
who(…)
{
    • • •
    amI();
    • • •
    amI();
    • • •
}
```

```
amI(…)
{
    •
    •
    amI();
    •
    •
}
```

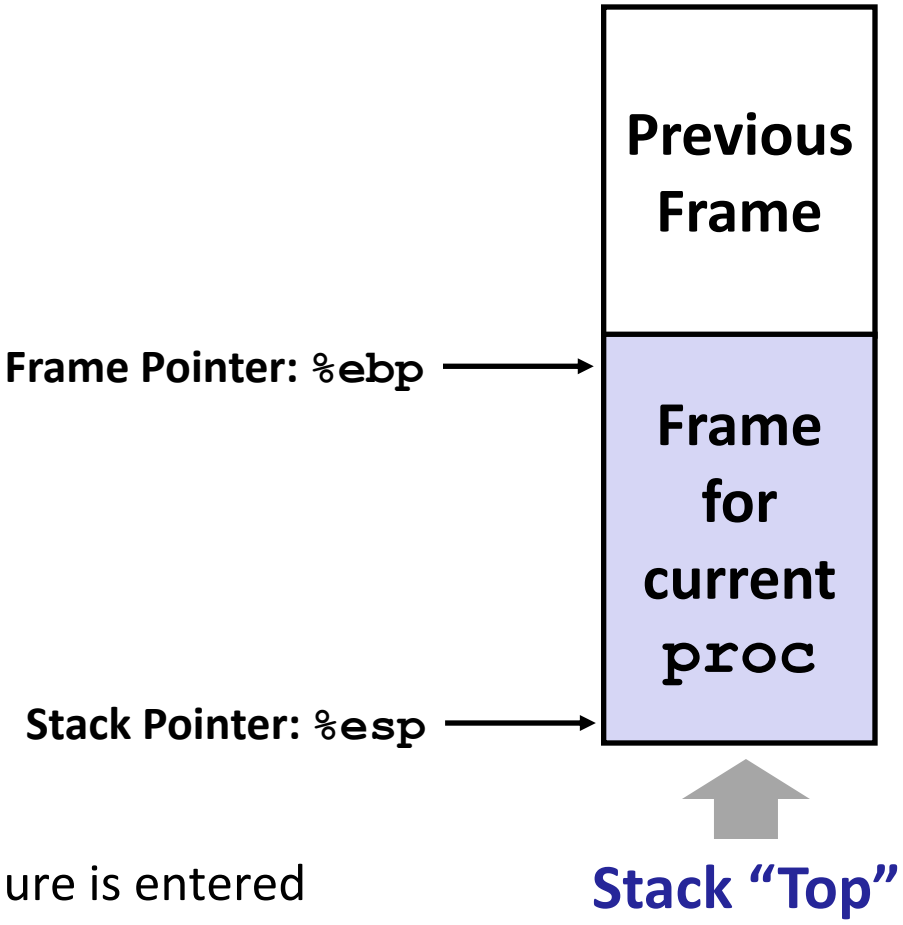
Example Call Chain



Procedure `amI` is recursive (calls itself)

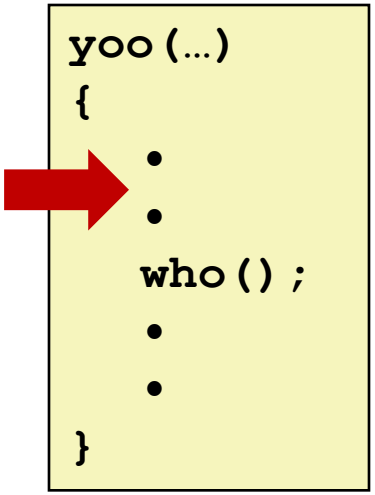
Stack Frames

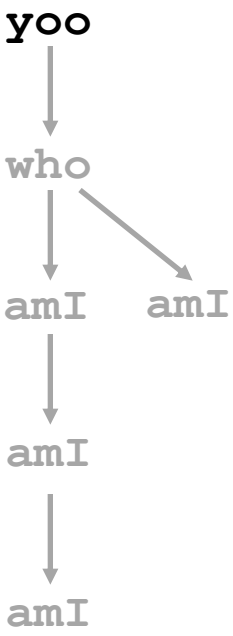
- **Contents**
  - Local variables
  - Function arguments
  - Return information
  - Temporary space

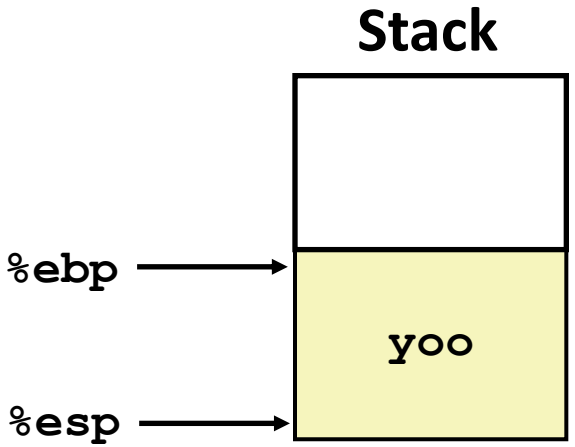


- **Management**
  - Space allocated when procedure is entered
    - "Set-up" code
  - Space deallocated upon return
    - "Finish" code

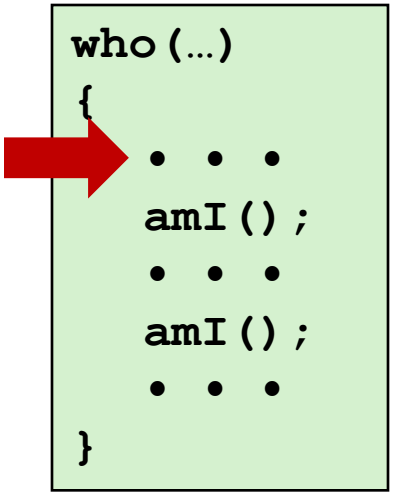
Example

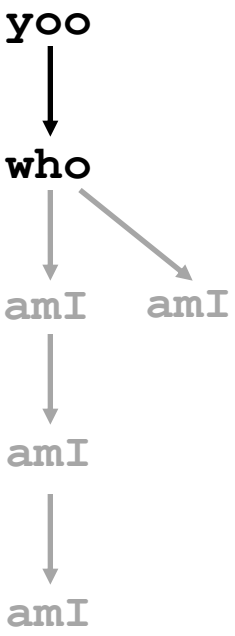


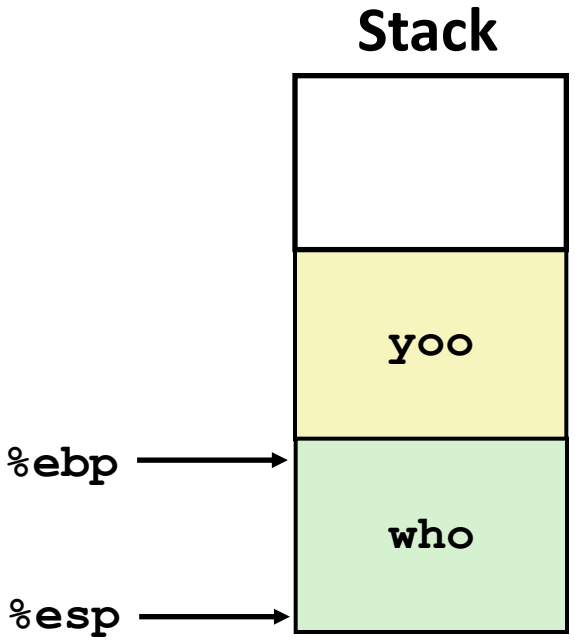




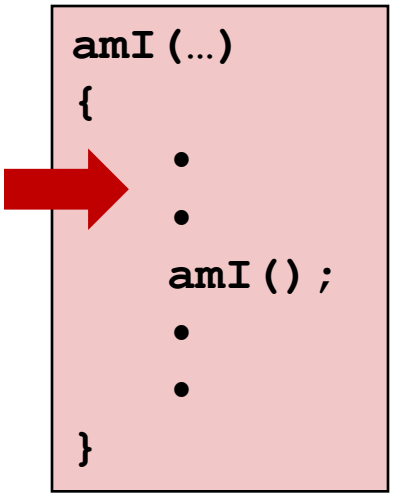
Example

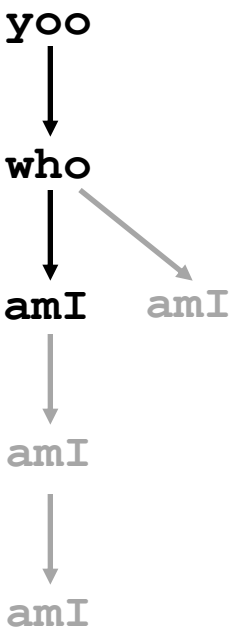


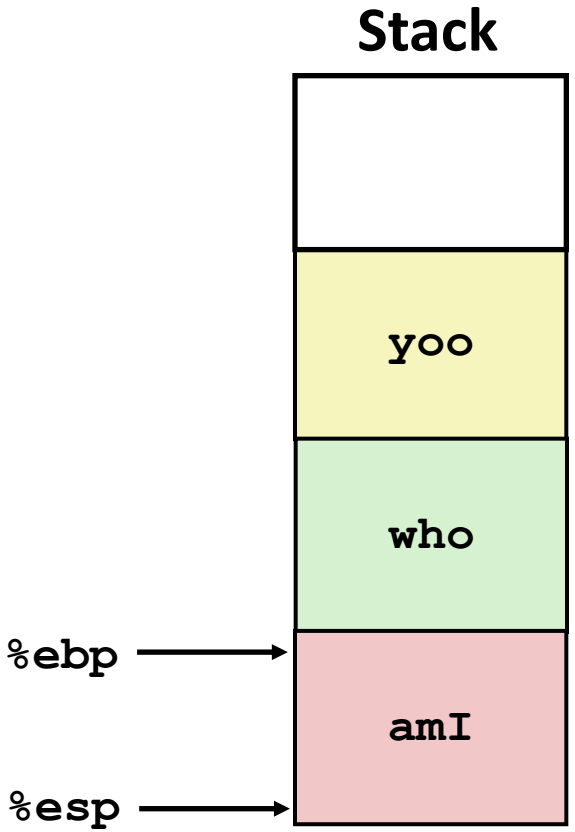




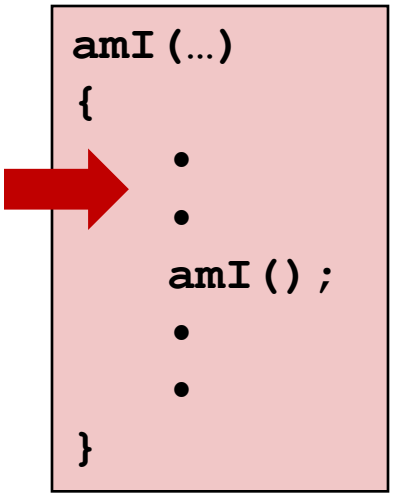
Example

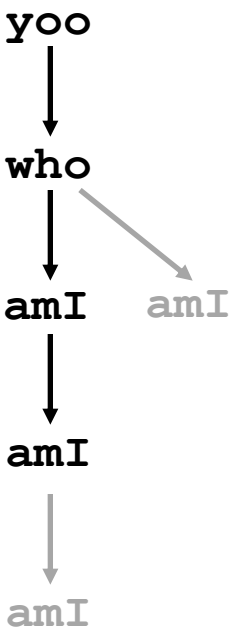


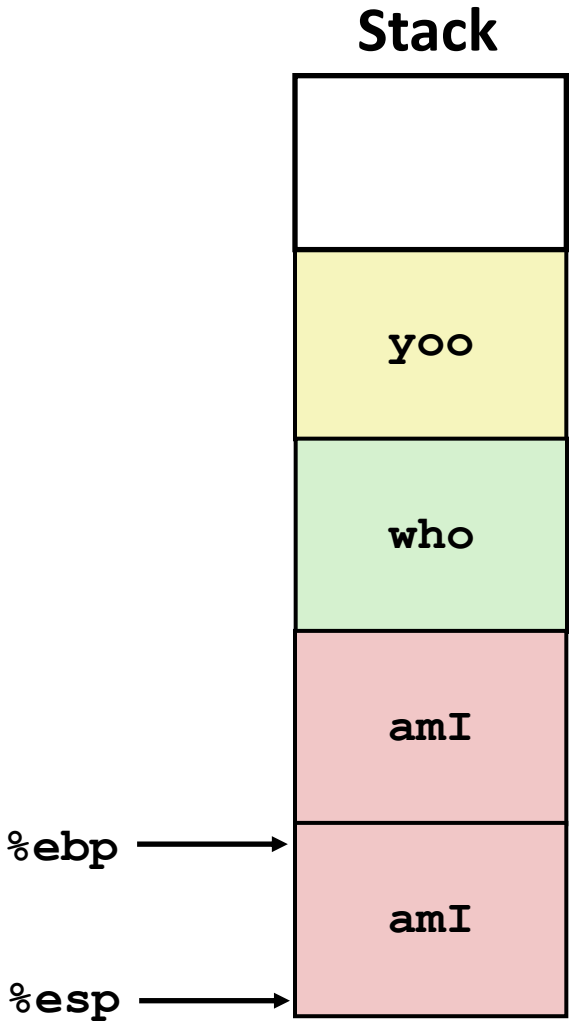




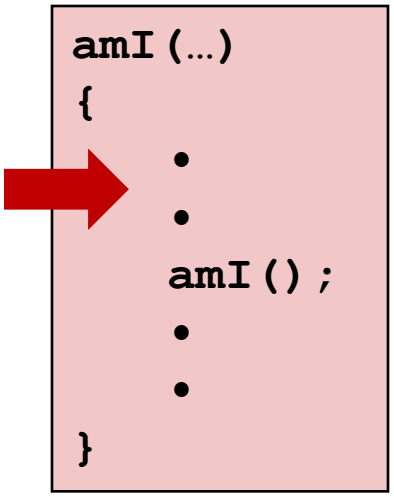
Example

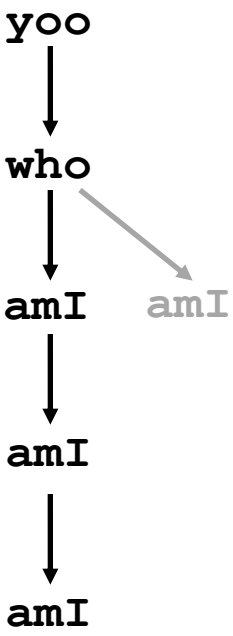


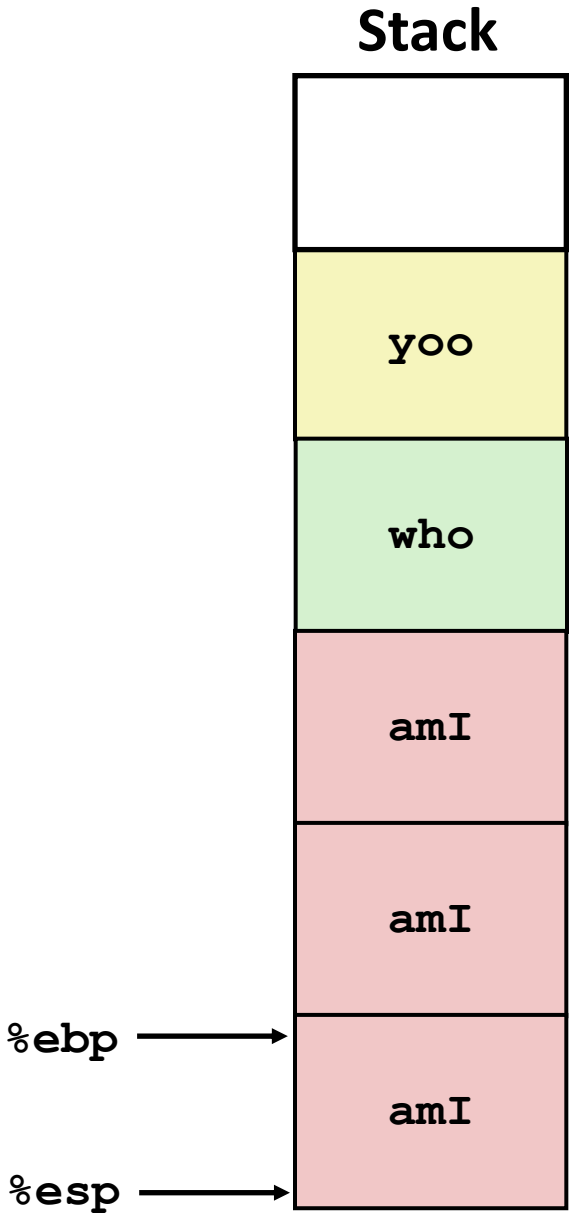




Example

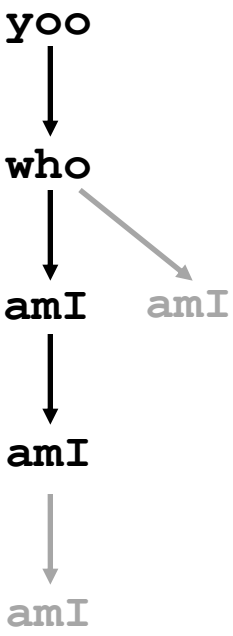

amI(…)
{
•
•
amI();
•
•
}

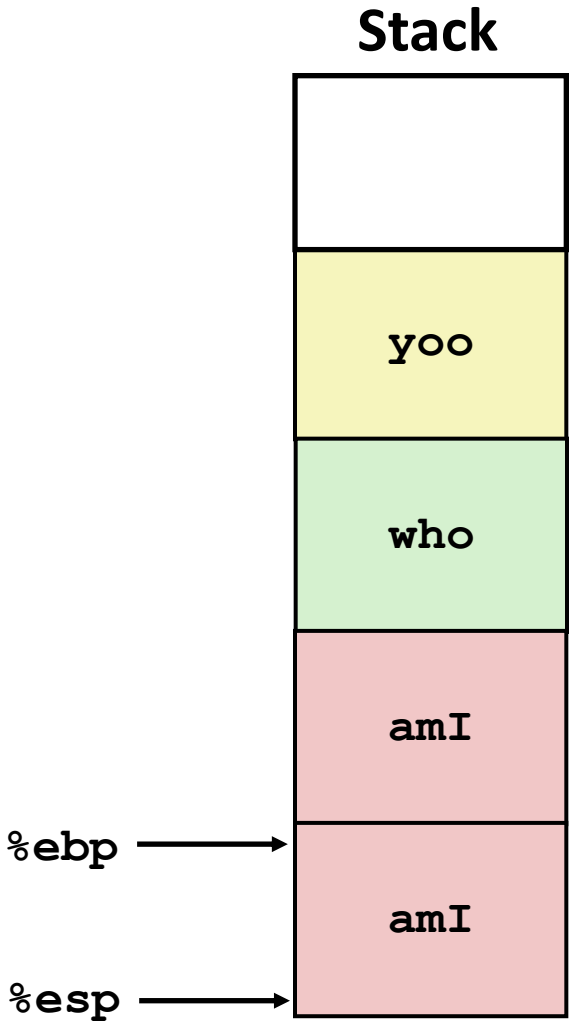

yoo
who
amI
amI
amI
amI


Stack
yoo
who
amI
amI
amI
%ebp
%esp

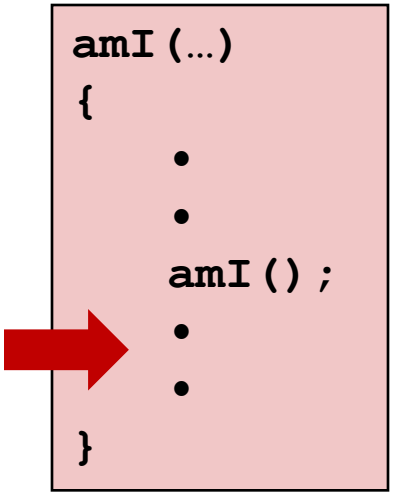
Example

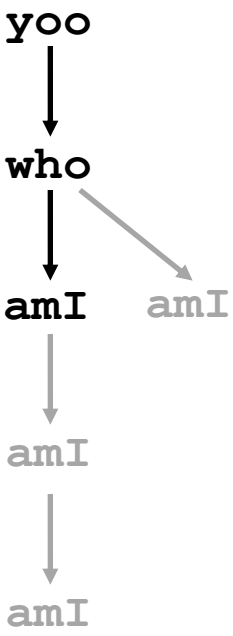
```
amI(...)
{
    •
    •
    amI();
    •
    •
}
```

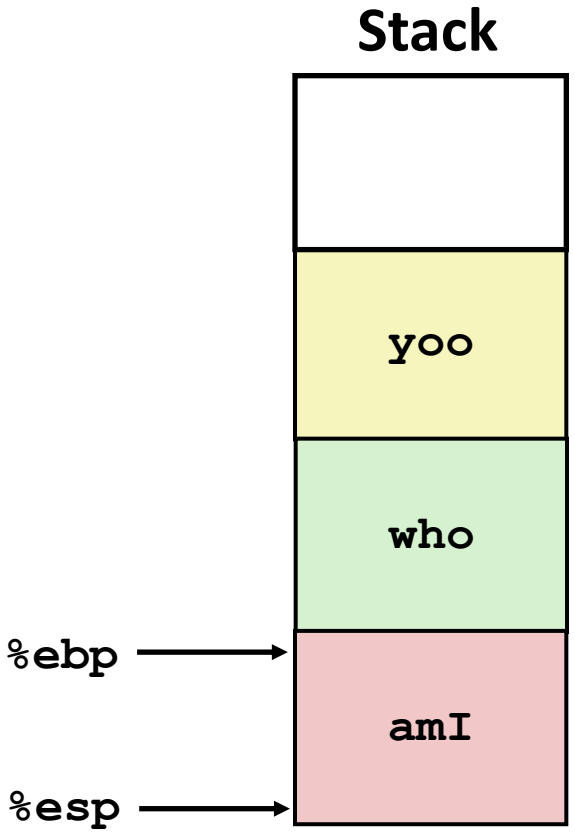




Example

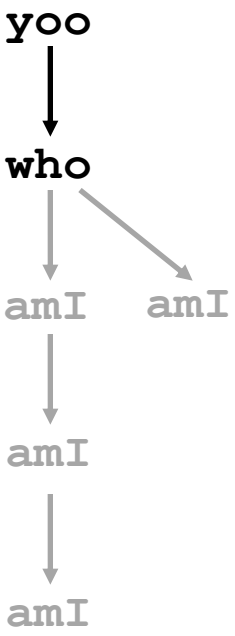


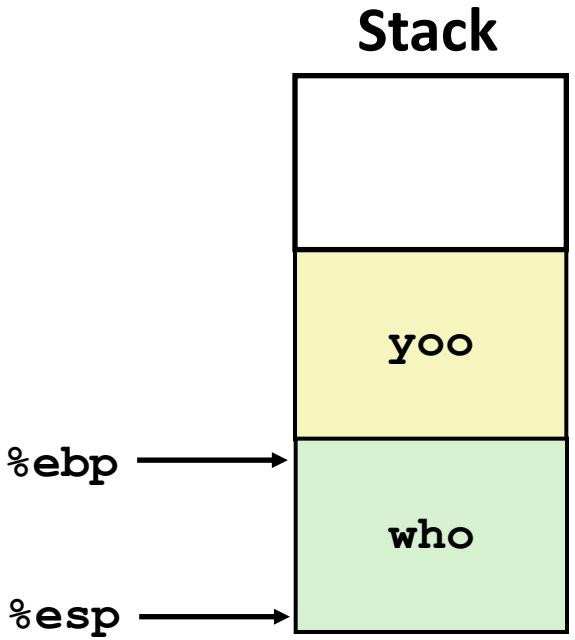




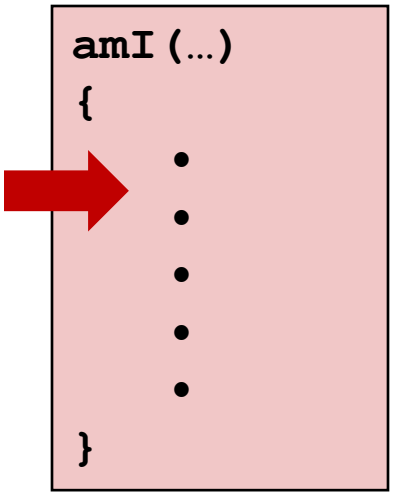
Example

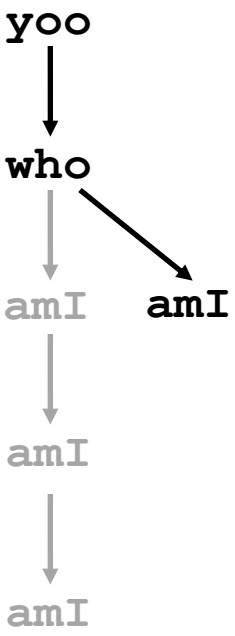
```
who(...)
{
    • • •
    amI();
    • • •
    amI();
    • • •
}
```

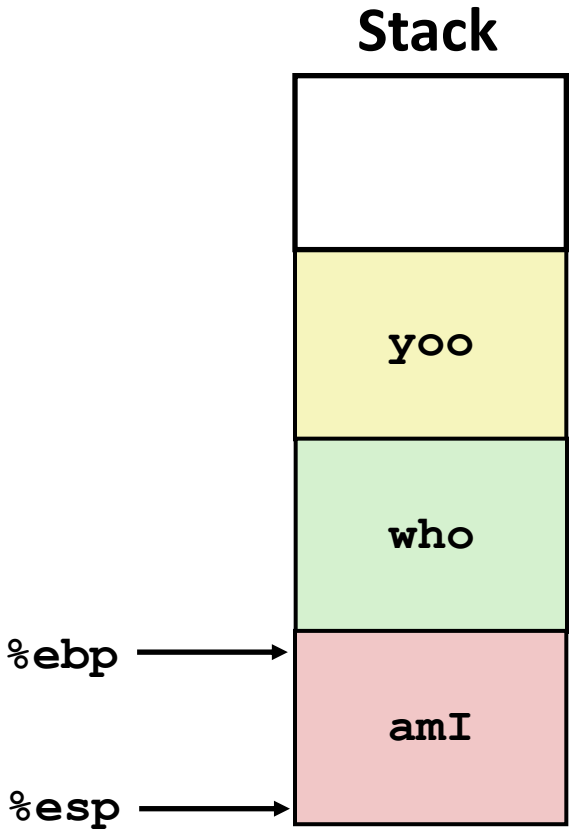




Example

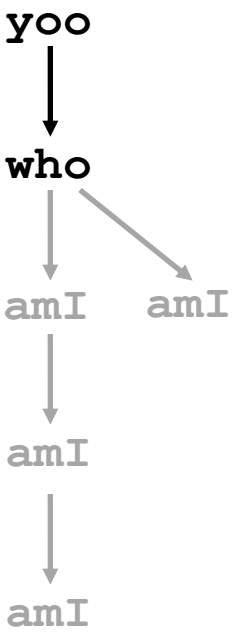


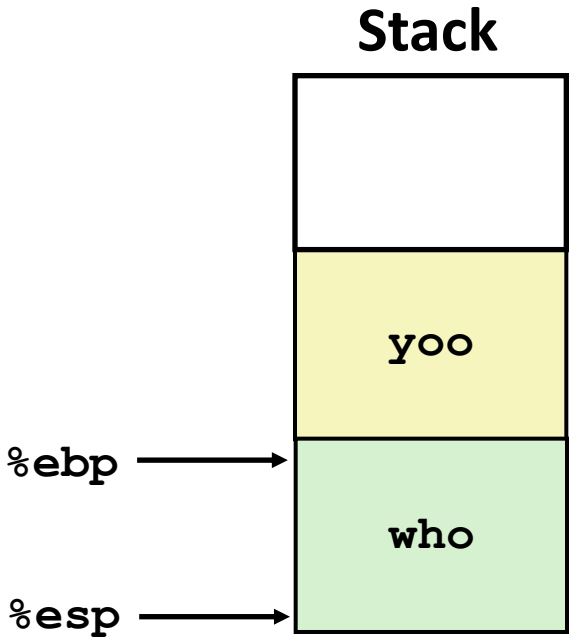




Example

```
who(…)
{
    • • •
    amI();
    • • •
    amI();
    • • •
}
```





Example

```
yoo(...)
{
    •
    •
    who();
    •
    •
}
```