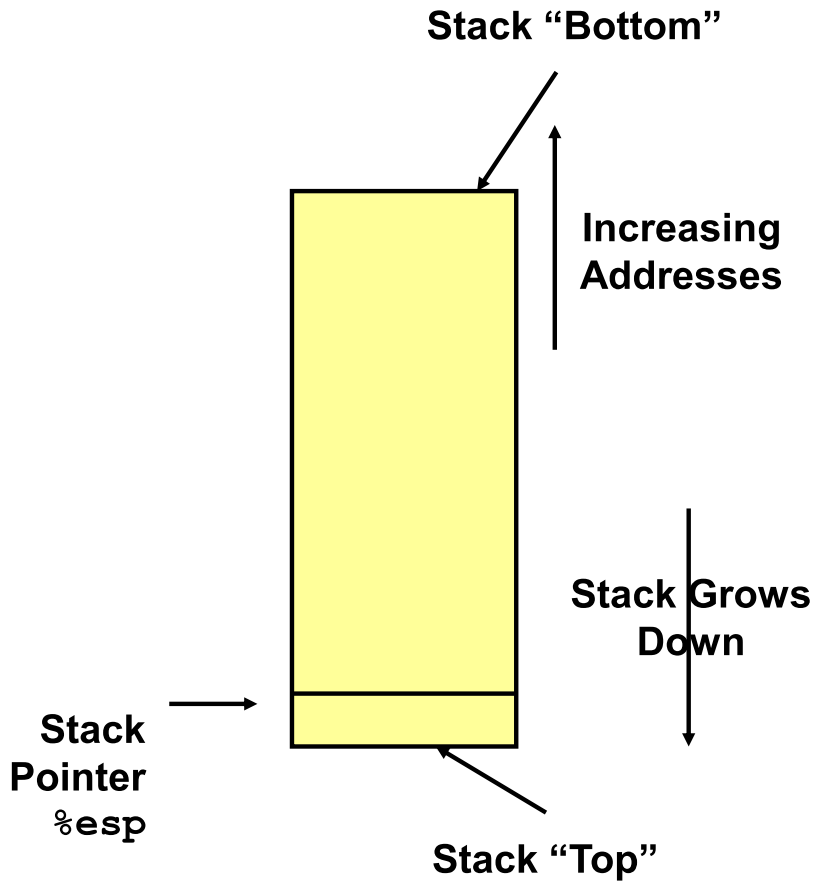
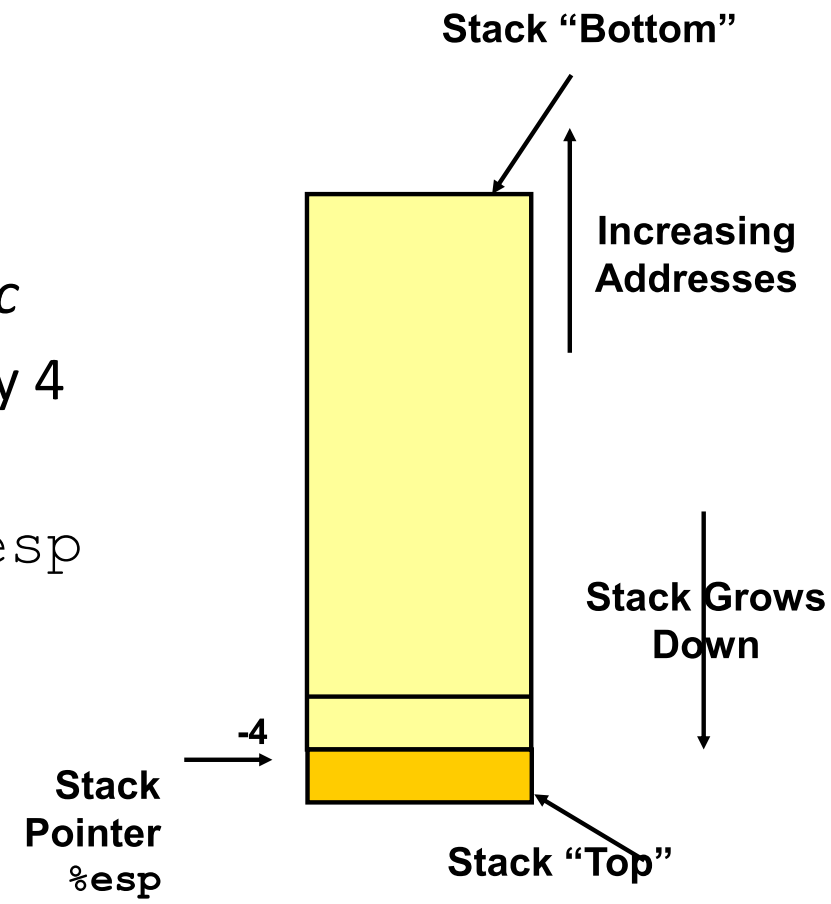
Intel Architecture 32-bit Stack

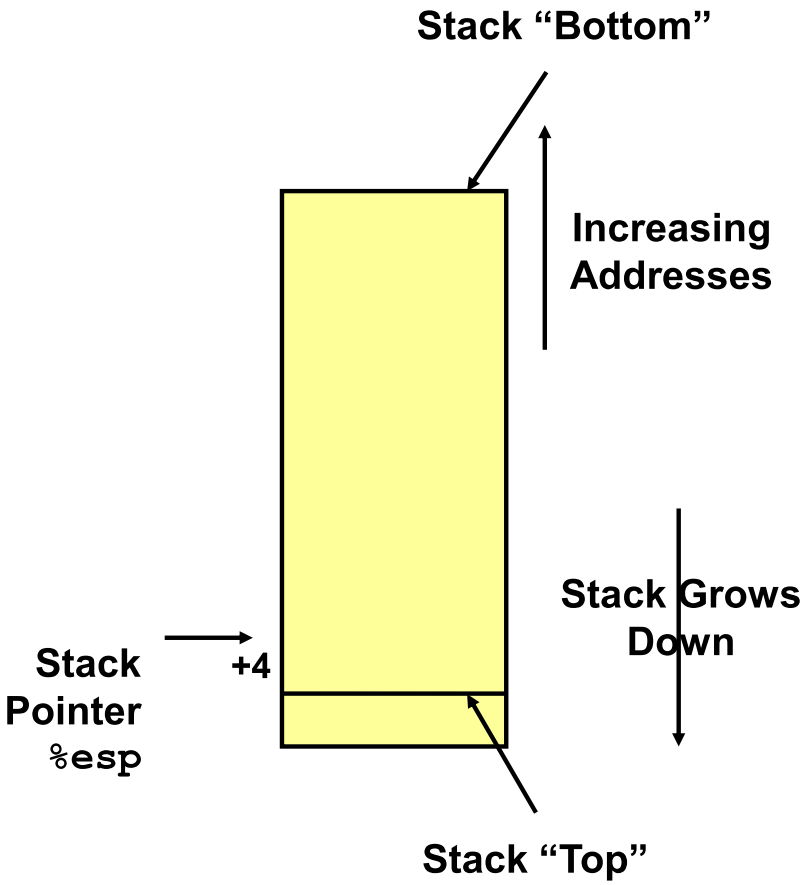
- Region of memory managed with stack discipline
- Grows toward lower addresses
- Register `%esp` indicates lowest stack address
  - address of top element



- Pushing
  - `pushl` *Src*
  - Fetch operand at *Src*
  - Decrement `%esp` by 4
  - Write operand at address given by `%esp`

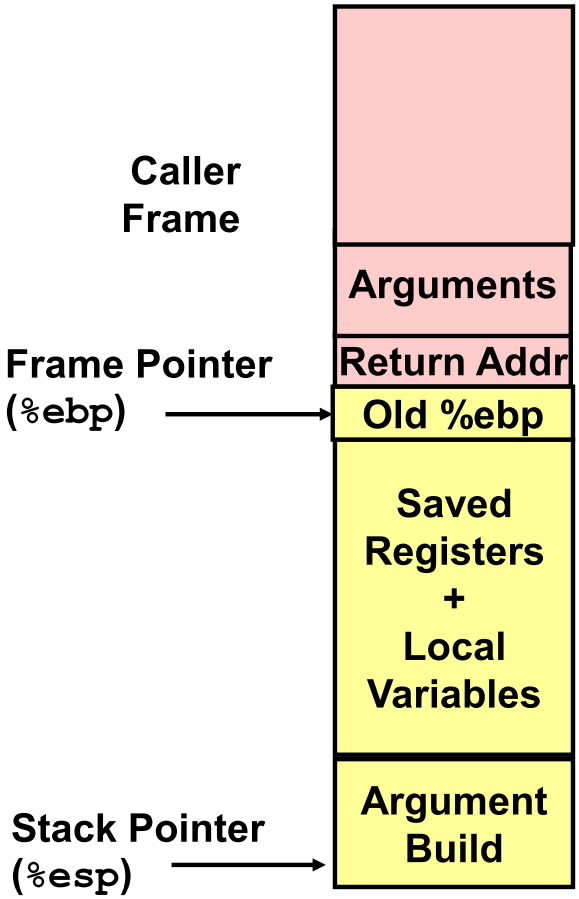


- Popping
  - `popl` *Dest*
  - Read operand at address given by `%esp`
  - Increment `%esp` by 4
  - Write to *Dest*



IA32/Linux Stack Frame

- Current Stack Frame ("Top" to Bottom)
  - Parameters for function about to call
    - "Argument build"
  - Local variables
    - If can't keep in registers
  - Saved register context
  - Old frame pointer
- Caller Stack Frame
  - Return address
    - Pushed by `call` instruction
  - Arguments for this call

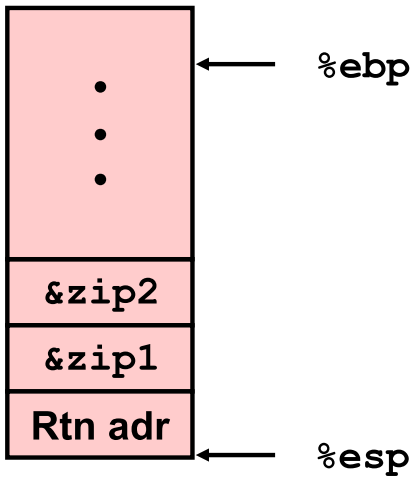


Example

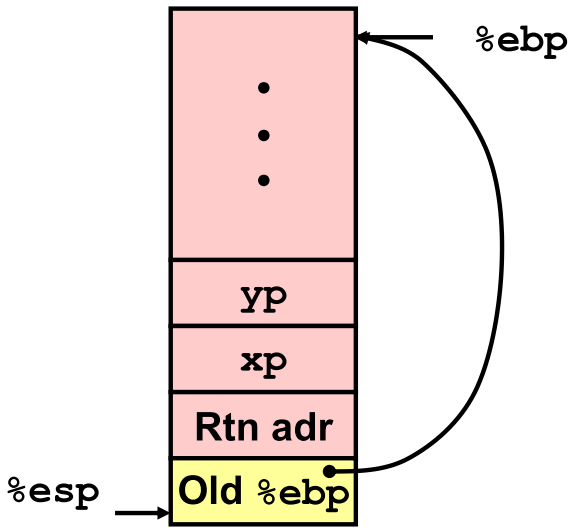
```
int zip1 = 15213;
int zip2 = 91125;

void call_swap()
{
  swap(&zip1, &zip2);
}
```

```
void swap(int *xp, int *yp)
{
  int t0 = *xp;
  int t1 = *yp;
  *xp = t1;
  *yp = t0;
}
```

```
swap:
    pushl %ebp
    movl %esp,%ebp
    subl $4, %esp
```



Resulting Stack

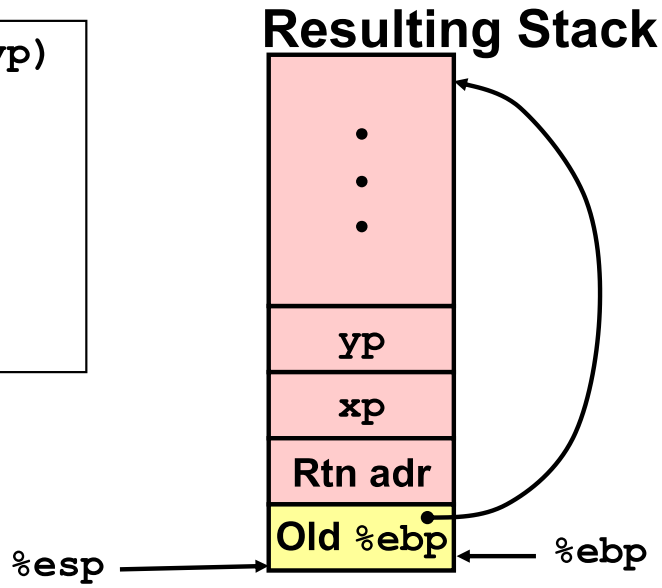


Example

```
int zip1 = 15213;
int zip2 = 91125;

void call_swap()
{
  swap(&zip1, &zip2);
}
```

```
void swap(int *xp, int *yp)
{
  int t0 = *xp;
  int t1 = *yp;
  *xp = t1;
  *yp = t0;
}
```

```
swap:
    pushl %ebp
    movl %esp,%ebp
    subl $4, %esp
```

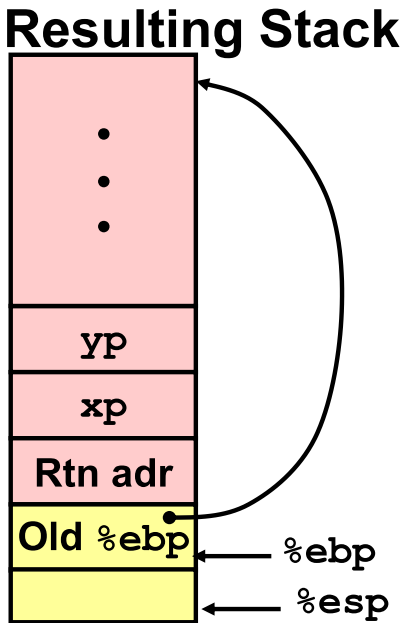


Example

```
int zip1 = 15213;
int zip2 = 91125;

void call_swap()
{
  swap(&zip1, &zip2);
}
```

```
void swap(int *xp, int *yp)
{
  int t0 = *xp;
  int t1 = *yp;
  *xp = t1;
  *yp = t0;
}
```

```
swap:
    pushl %ebp
    movl %esp,%ebp
    subl $4, %esp
```

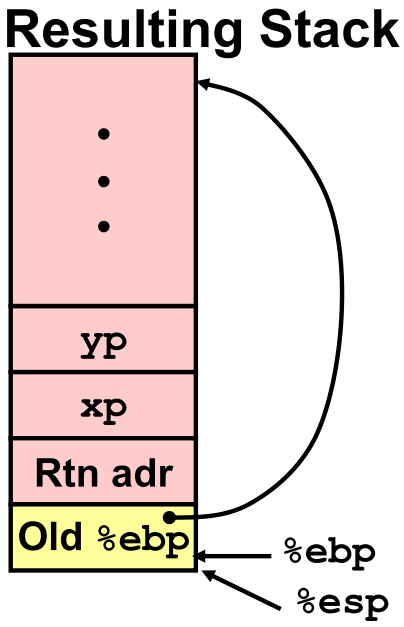


Example

```
int zip1 = 15213;
int zip2 = 91125;

void call_swap()
{
  swap(&zip1, &zip2);
}
```

```
void swap(int *xp, int *yp)
{
  int t0 = *xp;
  int t1 = *yp;
  *xp = t1;
  *yp = t0;
}
```

```
swap:
.....
    movl %ebp,%esp
    popl %ebp
    ret
```

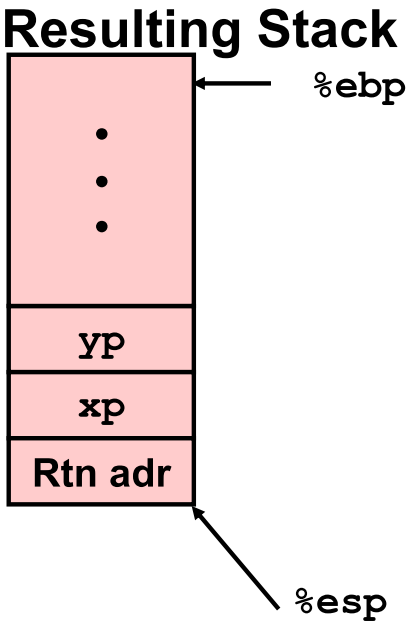


Example

```
int zip1 = 15213;
int zip2 = 91125;

void call_swap()
{
  swap(&zip1, &zip2);
}
```

```
void swap(int *xp, int *yp)
{
  int t0 = *xp;
  int t1 = *yp;
  *xp = t1;
  *yp = t0;
}
```

```
swap:
.....
    movl %ebp,%esp
    popl %ebp
    ret
```

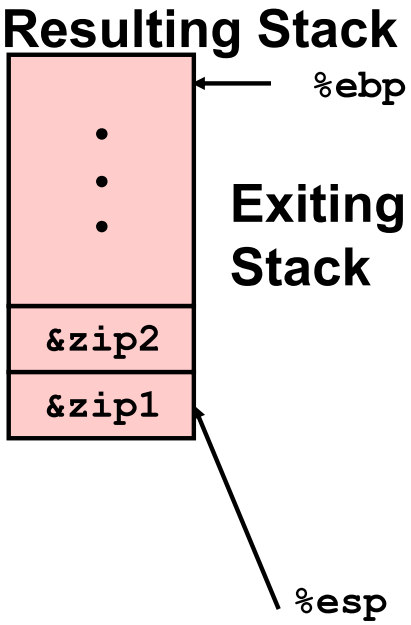


Example

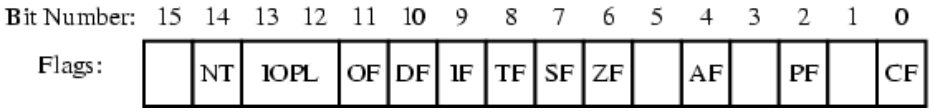
```
int zip1 = 15213;
int zip2 = 91125;

void call_swap()
{
  swap(&zip1, &zip2);
}
```

```
void swap(int *xp, int *yp)
{
  int t0 = *xp;
  int t1 = *yp;
  *xp = t1;
  *yp = t0;
}
```

```
swap:
.....
    movl %ebp,%esp
    popl %ebp
    ret
```



Flag register



AF -- Auxiliary Carry Flag

IOPL -- I/O Privilege Level (286+)

TF -- Trap Flag

PF -- Parity Flag

OF -- Overflow Flag

SF -- Sign Flag

CF -- Carry Flag

DF -- Direction Flag

ZF -- Zero Flag

Flags

- **CF** Carry Flag – becomes 1 if an addition, multiplication, AND, OR, etc results in a value larger than the register meant for the result
- **ZF** Zero Flag – becomes 1 if an operation results in a 0
- **SF** Sign Flag – is 1 if the value saved is negative, 0 for positive
- **OF** Overflow Flag – becomes 1 if the operation is larger than available space to write (e.g., addition which results in a number >32-bits)