

Low level attacks Assembly (part 2)

Mario Alviano

University of Calabria, Italy

A.Y. 2019/2020

Stack instructions

- `push operand`
- `pop address/register`
- Used for local variables
- Used to create cached copies
- Used for passing arguments to procedures

Load effective address

- Use `lea register, memory` to load a memory address into a register

Alert!

The use of square brackets in this case does not dereference!

Example

- `lea edi, [ebx+4*esi]`
- `lea eax, [var]`

Loops

- Conditional jumps can be used for implementing loops

```
MOV     CL, 10
L1:
<LOOP-BODY>
DEC     CL
JNZ     L1
```

- Alternatively, `loop label` can be used
- It decrements `ECX` and jumps to `label` if not zero

```
mov ECX, 10
l1:
<loop body>
loop l1
```

Example

Try `loops.asm`

- Subroutines are identified by labels
- Subroutines are called by `call label`
 - Pushes EIP into the stack, and jumps to `label`
- Each subroutine terminates with `ret`
 - Pops an address from the stack, and jumps to it

Example

Try `subroutine.asm`

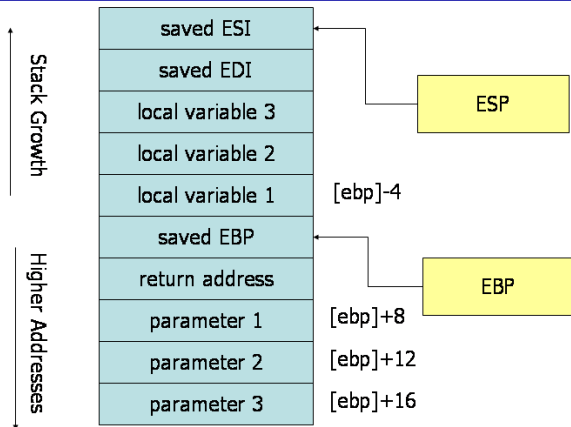
Calling convention

- How to share subroutines?
- We must agree on some strategy to pass parameters
- Several conventions do exist
- We will consider the C/C++ convention
- Essentially, use the stack!

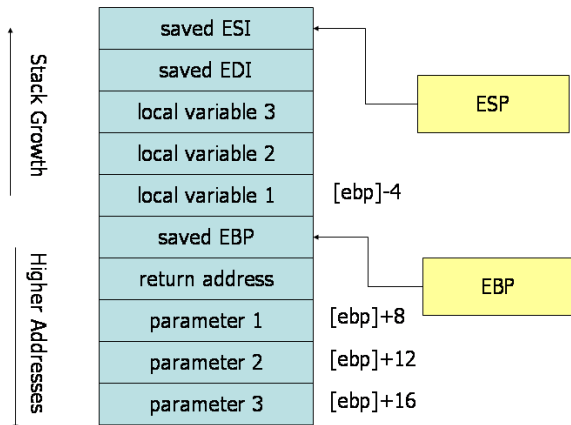
Two sets of rules

- 1 The first set is for the caller
- 2 The second set is for the callee

Caller rules

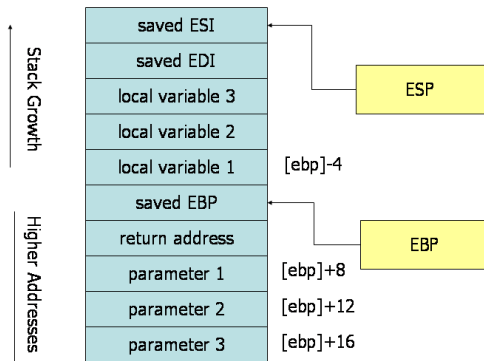


- 1 Push *caller-saved* registers: EAX, ECX, EDX
- 2 Push arguments in reverse order (allow varadics)
- 3 Use the `call` instruction (push return address, and jump)
- 4 Remove parameters from the stack (add their size to ESP)
- 5 Restore caller-saved registers (pop them from the stack)



Subroutine Prologue

- 1 Push EBP, and then copy ESP into EBP
 - All parameters are in EBP-offset
- 2 Allocate local variables in the stack
 - Subtract their size from ESP
 - All local variables are in EBP+offset
- 3 Push *callee-saved* registers: EBX, EDI, ESI



Subroutine Epilogue

- 1 Leave the return value in EAX
- 2 Restore callee-saved registers (pop them)
- 3 Deallocate local variables
 - Add their size to ESP
 - Better alternative, copy EBP into ESP
- 4 Restore the previous EBP (pop it)
- 5 Return to the caller by executing `ret`

Instruction `leave` is equivalent to

- `mov esp, ebp`
- `pop ebp`

It is a shortcut for **3** and **4** in the previous slide.

Example

Try `convention.asm`

Integration with the C/C++ libraries

- Declare used functions: `extern printf`
- The entry point is the function `main`
- Use everything we just learned about assembly!

Example

Write and read

- Use `printf` to write in STDOUT
- Use `scanf` to read from STDIN
- Let's have a look at `printf.asm` and `scanf.asm`

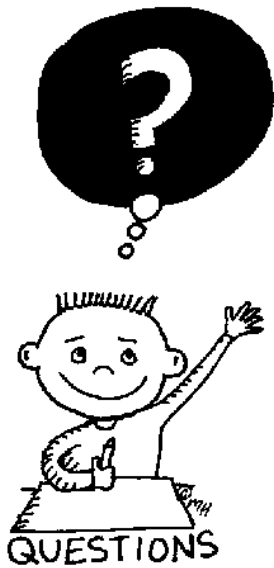
Exercises

Read a sequence of positive integers terminated by -1, and

- 1 Print the maximum number of the sequence
- 2 Print the sum of all numbers
- 3 Print the sum of all even numbers
- 4 Print the size of the largest subsequence of even numbers

Read a sequence of N integers (N read from STDIN), and

- 5 Print 1 if the sequence is a palindrome, otherwise print 0
- 6 Print the most frequent number
- 7 Print the less frequent number



END OF THE
LECTURE