Real-world architectures

The call stack

A **stack** is a data structure for accessing data in LIFO (last in, first out) order.

Data items can be **pushed** onto a stack, and **popped** off. Example in pseudocode:

```
push(3)
push(-5)
push(8)
x = pop()
push(4)
y = pop()
z = pop()
```

This results in `x=8, y=4, z=-5`.

In modern computers, every running program (and actually every thread) has a **call stack**, also known as "the stack." Details vary, but typical usage is that on every method call, the following information is pushed onto the stack:

- parameters
- return address (i.e. where the program should continue after the method terminates)
- local variables

This collection of data is the method's **frame** or **stack frame**.

When the method returns, the data in this stack frame is popped off the stack and execution continues from the popped return address. (Note that this approach is different from MARIE's way of calling subroutines. The above approach is more complex but also more beneficial because it permits recursion.)

The stack of methods you see in a debugger is derived from the call stack. (Example: execute StackDemo.java in Eclipse with a breakpoint enabled, and observe the call stack.)

A few details about three real architectures

Architectures are often classified by their word size (e.g. 16-bit, 32-bit, 64-bit). The word size is the number of bits in the registers of an architecture, and usually the number of bits that can be simultaneously processed in a single operation. For example, in a 32-bit architecture, most of the registers hold 32 bits and the architecture is capable of adding or multiplying two 32-bit registers in a single cycle. We will learn a few details about the following three architectures: IA32, x64, MIPS.

1. IA32

Intel's 32-bit architecture known as IA32 or x86 is an important real-world architecture, popular from the 1980s to the 2000s. IA32 is the basis for the famous chips and chip families such as the 80386, x86, Pentium, and Core.

2. x64 (also known as x86-64, AMD64, Intel 64)

This is a very common architecture in today's computers. It is a 64-bit architecture, and is the basis for many current chips including Opteron , Athlon, Core i7, Celeron D, Pentium D.

The x64 architecture includes many predefined registers such as:

- 16 general-purpose registers with names like RAX, RBX, R8, R9. The general-purpose registers include some registers that are almost always used for particular jobs, such as:
    - RSP -- the stack pointer, which holds the address of the top of the stack
    - RBP -- the base pointer, which holds the address of the bottom of the current stack frame
- a special register RIP, which serves as the program counter (also known as the instruction pointer)

The x64 architecture includes many useful instructions, such as

- ADD, MULTIPLY, LOAD, STORE, JUMP. These can all be activated in various modes such as immediate, direct, and indirect (these modes will be defined in a future lecture)
- a CALL instruction for invoking methods
- a LOOP instruction that increments a counter and jumps to the top of the loop
- atomic increment and decrement instructions
- many instructions for operating system functionality, for example: interrupts, switching between processes, memory management, security and protection

Demo/experiment: compile variables.c, loop.c, subroutine.c -- look at the resulting assembly language

- to produce the assembly code, use "gcc –S variables.c"
- edit the program in a text editor and regenerate the assembly, try to note any differences

3. MIPS

This is a famous and highly regarded architecture but seems to have fallen by the wayside in recent years. It was used in the original Tivo and in the PlayStation I and II models. Interesting features include:

- a large number of general-purpose registers (32 of them)
- it is a **load/store architecture**, meaning every instruction (except load and store themselves) use registers as operands. So we never see memory locations as operands, except for load and store.