# COMP 222



Rev 3-14-22

## Computer Organization (Architecture)

# Stack Machines

Dr Jeff Drobman

website → drjeffsoftware.com/classroom.html

email → jeffrey.drobman@csun.edu

# Stack Machines



Quora

## What is better: a stack-based machine or a register-based machine?

**Jeff Drobman**, Lecturer at California State University, Northridge (2016-present)

Answered just now

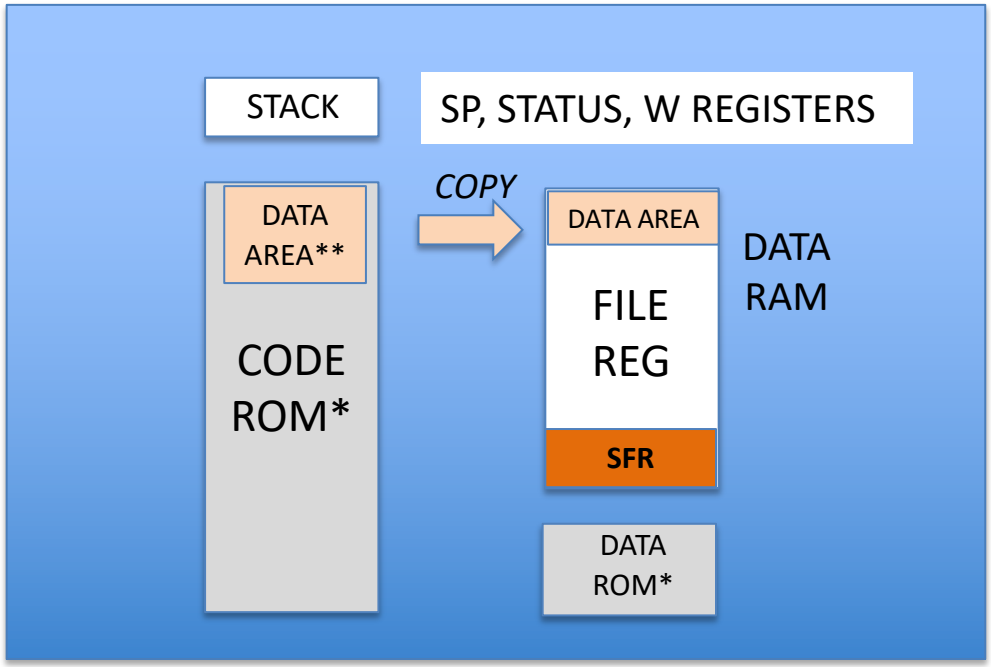
a “General Register” RISC architecture wins by having a large register file, all of which can be used as operands and results for data, and addresses as pointers. this is better than a “dedicated register” architecture like x86. stacks use data memory (mostly L1 D-cache), which takes additional cycles to access, plus access is limited to the top of the stack.

# PIC18F Dual Memory





## PIC 18F MICROCONTROLLER




*READ ONLY

**use **DB, DW**

# Stack Machines

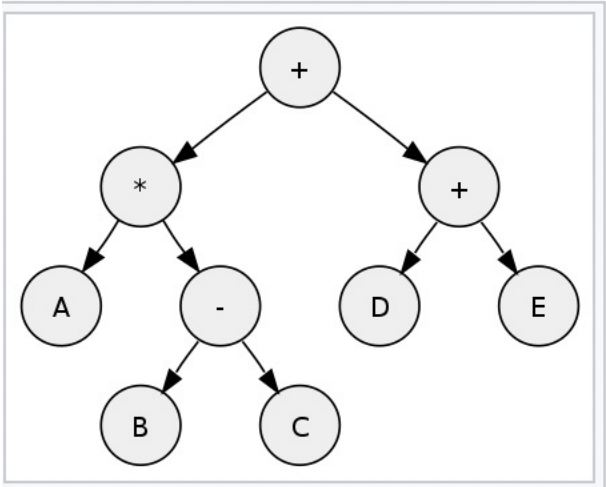


## Stack machine

From Wikipedia, the free encyclopedia

**Reverse Polish notation** (**RPN**), also known as **Polish postfix notation** or simply **postfix notation**, is a mathematical notation in which operators *follow* their operands, in contrast to Polish notation (PN), in which operators *precede* their operands. It does not need any parentheses as long as each operator ha

- ***APL, Forth*** languages
- HP, TI calculators




Binary syntax tree for the expression *A**(*B*-*C*) + (*D*+*E*)

# Stack Machines



## Stack machine

From Wikipedia, the free encyclopedia

For example, consider the expression *A**(*B*-*C*)+(*D*+*E*), written in reverse Polish notation as *A B C* - * *D E* + +. Compiling and running this on a simple imaginary stack machine would take the form:

```
                # stack contents (leftmost = top = most recent):
push A          #           A
push B          #     B     A
push C          # C   B     A
subtract        #     B-C   A
multiply        #           A*(B-C)
push D          #     D     A*(B-C)
push E          # E   D     A*(B-C)
add             #     D+E   A*(B-C)
add             #           A*(B-C)+(D+E)
```

- Hardware or Software stacks

# Stack Machines



## Stack machine

From Wikipedia, the free encyclopedia

In computer science, computer engineering and programming language implementations, a **stack machine** is a computer processor or a virtual machinein which the primary interaction is moving short-lived temporary values to and from a push down stack. In the case of a hardware processor, a hardware stack is used. The use of a stack significantly reduces the required number of processor registers. Stack machines extend push-down automaton with additional load/store operations or multiple stacks and hence are Turing-complete.

- Turing Complete

In computability theory, a system of data-manipulation rules is said to be **Turing-complete** or **computationally universal** if it can be used to simulate any Turing machine. This means that this system is able to recognize or decide other data-manipulation rule sets. Turing completeness is used a

# Stack Machines



## Stack machine

From Wikipedia, the free encyclopedia

Most or all stack machine instructions assume that operands will be from the stack, and results placed in the stack. The stack easily holds more than two inputs or more than one result, so a rich set of operations can be computed. In stack machine code (sometimes called p-code), instructions will frequently have only an opcode commanding an operation, with no additional fields identifying a constant, register or memory cell, known as a **zero address format**.[1] This greatly simplifies instruction decoding. Branches, load immediates, and load/store instructions require an argument field, but stack machines often arrange that the frequent cases of these still fit together with the opcode into a compact group of bits. The selection of operands from prior results is done implicitly by ordering the instructions. Some stack machine instruction sets are intended for interpretive execution of a virtual machine, rather than driving hardware directly.

- Zero address format

# Stack Machines



## Stack machine

From Wikipedia, the free encyclopedia

**Commercial stack machines**[edit]

*See also: High-level language computer architecture*

Examples of stack instruction sets directly executed in hardware include

- The F18A architecture of the 144-processor GA144 chip from GreenArrays, Inc.[7][8][9]
- the Z4 computer by Konrad Zuse.[10]
- the Burroughs large systems architecture (since 1961)
- the Xerox Dandelion introduced April 27, 1981, utilized a stack machine architecture to save memory.[11][12]
- the English Electric KDF9 machine. First delivered in 1964, the KDF9 had a 19-deep pushdown stack of arithmetic registers, and a 17-deep stack for subroutine return addresses
- the UCSD Pascal p-machine (as the Pascal MicroEngine and many others) supported a complete student programming environment on early 8-bit microprocessors with poor instruction sets and little RAM, by compiling to a virtual stack machine.
- HP 3000 (Classic, not PA-RISC)
- Tandem Computers T/16. Like HP 3000, except that compilers, not microcode, controlled when the register stack spilled to the memory stack or was refilled from the memory stack.
- the Atmel MARC4 microcontroller[14]
- Several "Forth chips"[15] such as the RTX2000, the RTX2010, the F21[16] and the PSC1000[17][18]

# Stack Machines



## Stack machine

From Wikipedia, the free encyclopedia

**Virtual stack machines**[edit]

Examples of virtual stack machines interpreted in software:

- the Whetstone ALGOL 60 interpretive code,[22] on which some features of the Burroughs B6500 were based
- the Burroughs B5000
- the UCSD Pascal p-machine; which closely resembled Burroughs
- the Niklaus Wirth p-code machine
- Smalltalk
- the Java virtual machine instruction set
- the WebAssembly bytecode
- the Virtual Execution System (VES) for the Common Intermediate Language (CIL) instruction set of the .NET Framework (ECMA 335)
- the Forth programming language, especially the integral virtual machine
- Adobe's PostScript

# Stack Machines



## Stack machine

From Wikipedia, the free encyclopedia

**Hybrid machines**[edit]

(These should not be confused with hybrid computers that combine both digital and analogue features, e.g. an otherwise digital computer that accesses analogue multiplication or differential equation solving by memory mapping and conversion to and from an analogue device's inputs and outputs.)

Pure **stack machines** are quite inefficient for procedures which access multiple fields from the same object. The stack machine code must reload the object pointer for each pointer+offset calculation. A common fix for this is to add some register-machine features to the stack machine: a visible **register file** dedicated to holding addresses, and register-style instructions for doing loads and simple address calculations. It is uncommon to have the registers be fully general purpose, because then there is no strong reason to have an expression stack and postfix instructions.

# Stack Machines



## Stack machine

From Wikipedia, the free encyclopedia

Another common hybrid is to start with a register machine architecture, and add another memory address mode which emulates the push or pop operations of stack machines: 'memaddress = reg; reg += instr.displ'. This was first used in DEC's PDP-11 minicomputer. This feature was carried forward in VAX computers and in Motorola 6800 and M68000 microprocessors. This allowed the use of simpler stack methods in early compilers. It also efficiently supported virtual machines using stack interpreters or threaded code. However, this feature did not help the register machine's own code to become as compact as pure stack machine code. Also, the execution speed was less than when compiling well to the register architecture. It is faster to change the top-of-stack pointer only occasionally (once per call or return) rather than constantly stepping it up and down throughout each program statement, and it is even faster to avoid memory references entirely.

# Stack Machines



## Stack machine

From Wikipedia, the free encyclopedia

**Stack machines** have higher code density. In contrast to common stack machine instructions which can easily fit in 6 bits or less, register machines require two or three register-number fields per ALU instruction to select operands; the densest register machines average about 16 bits per instruction plus the operands. Register machines also use a wider offset field for load-store opcodes. A **stack machine's compact code** naturally fits more instructions in **cache**, and therefore could achieve better cache efficiency, reducing memory costs or permitting faster memory systems for a given cost. In addition, most stack-machine instruction is very simple, made from only one opcode field or one operand field. Thus, stack-machines require very little electronic resources to decode each instruction.

A program has to **execute more instructions** when compiled to a **stack machine** than when compiled to a register machine or memory-to-memory machine

# Stack Machines



## Stack machine

From Wikipedia, the free encyclopedia

More recently, so-called **second-generation stack machines** have adopted a **dedicated** collection of registers to serve as address registers, off-loading the task of memory addressing from the data stack.

For example, MuP21 relies on a register called "A", while the more recent GreenArrays processors relies on two registers: A and B.

# Stack Machines



## Stack machine

From Wikipedia, the free encyclopedia

x87

The Intel x86 family of microprocessors have a register-style (**accumulator**) instruction set for most operations, but use **stack instructions** for its x87, Intel 8087 floating point arithmetic, dating back to the iAPX87 (**8087**) coprocessor for the 8086 and 8088. That is, there are no programmer-accessible floating point registers, but only an **80-bit wide, 8 deep stack**. The x87 relies heavily on the x86 CPU for assistance in performing its operations.

# Stack Machines



## Stack machine

From Wikipedia, the free encyclopedia

**Interrupts**

Responding to an interrupt involves saving the registers to a stack, and then branching to the interrupt handler code. Often stack machines respond more quickly to interrupts, because most parameters are already on a stack and there is no need to push them there. Some register machines deal with this by having multiple register files that can be instantly swapped but this increases costs and slows down the register file.

# Stack Machines



## Stack machine

From Wikipedia, the free encyclopedia

**Out-of-order execution**[edit]

This view permits the out-of-order execution of the Tomasulo algorithm to be used with stack machines.

Out-of-order execution in stack machines seems to reduce or avoid many theoretical and practical difficulties.[31] The cited research shows that such a stack machine can exploit **instruction-level parallelism**, and the resulting hardware must **cache data** for the instructions. Such machines effectively bypass most memory accesses to the stack. The result achieves **throughput** (instructions per clock) **comparable to** RISC register machines, with much **higher code densities** (because operand addresses are implicit).

One issue brought up in the research was that it takes about **1.88** stack-machine instructions to do the work of a register machine's RISC instruction. Competitive out-of-order stack machines therefore require about twice as many electronic resources to track instructions ("issue stations"). This might be compensated by savings in instruction cache and memory and instruction decoding circuits.