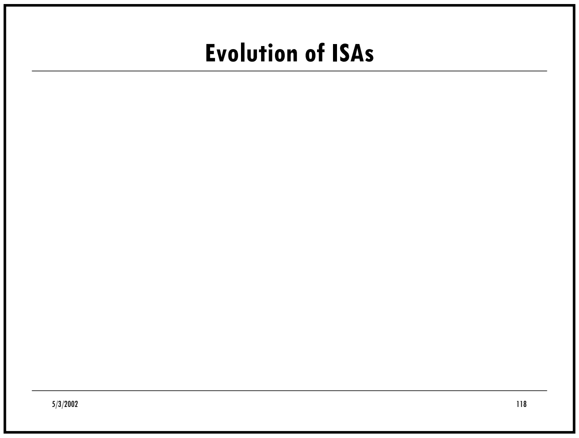
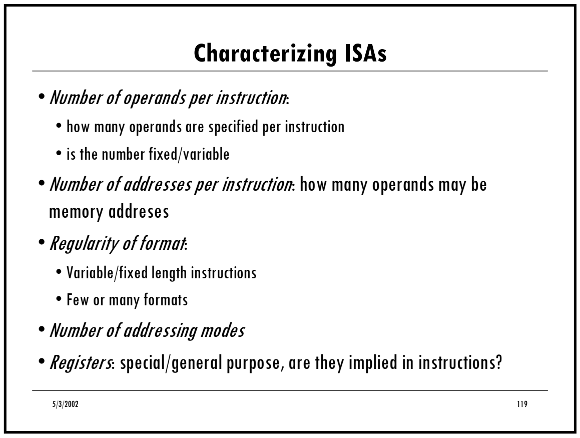
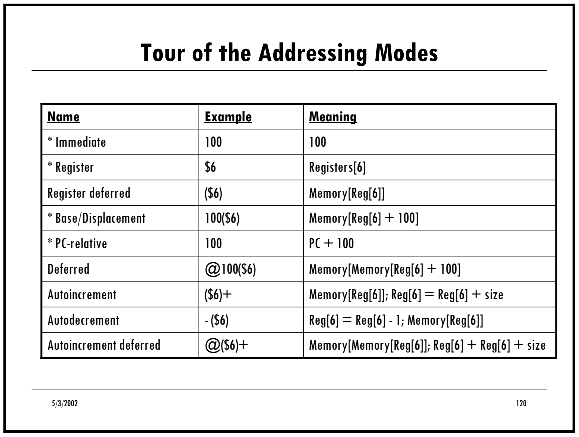


Evolution of ISAs



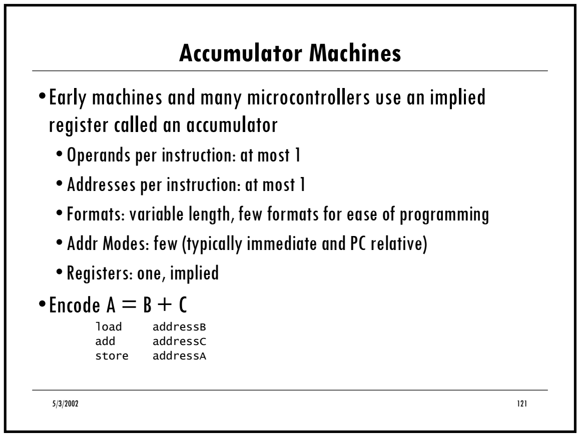
Characterizing ISAs

- *Number of operands per instruction*:
  - how many operands are specified per instruction
  - is the number fixed/variable
- *Number of addresses per instruction*: how many operands may be memory addreses
- *Regularity of format*:
  - Variable/fixed length instructions
  - Few or many formats
- *Number of addressing modes*
- *Registers*: special/general purpose, are they implied in instructions?



Tour of the Addressing Modes

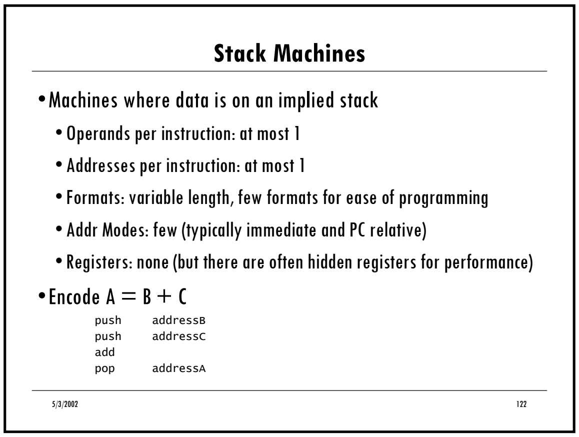
Name	**Example**	**Meaning**
* Immediate	100	100
* Register	$6	Registers[6]
Register deferred	($6)	Memory[Reg[6]]
* Base/Displacement	100($6)	Memory[Reg[6] + 100]
* PC-relative	100	PC + 100
Deferred	@100($6)	Memory[Memory[Reg[6] + 100]
Autoincrement	($6)+	Memory[Reg[6]]; Reg[6] = Reg[6] + size
Autodecrement	- ($6)	Reg[6] = Reg[6] - 1; Memory[Reg[6]]
Autoincrement deferred	@($6)+	Memory[Memory[Reg[6]]; Reg[6] + Reg[6] + size



Accumulator Machines

- Early machines and many microcontrollers use an implied register called an accumulator
  - Operands per instruction: at most 1
  - Addresses per instruction: at most 1
  - Formats: variable length, few formats for ease of programming
  - Addr Modes: few (typically immediate and PC relative)
  - Registers: one, implied
- Encode A = B + C

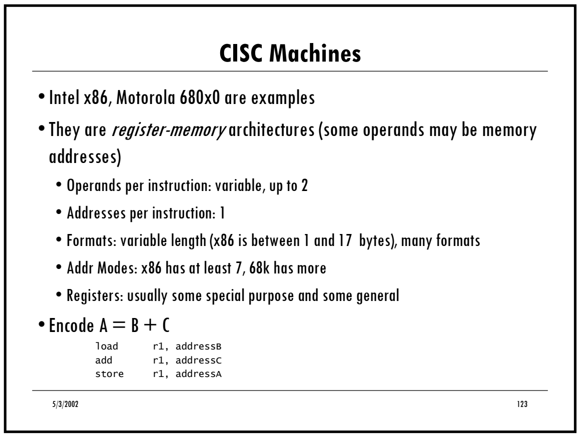
```
load     addressB
add      addressC
store    addressA
```



Stack Machines

- Machines where data is on an implied stack
  - Operands per instruction: at most 1
  - Addresses per instruction: at most 1
  - Formats: variable length, few formats for ease of programming
  - Addr Modes: few (typically immediate and PC relative)
  - Registers: none (but there are often hidden registers for performance)
- Encode A = B + C

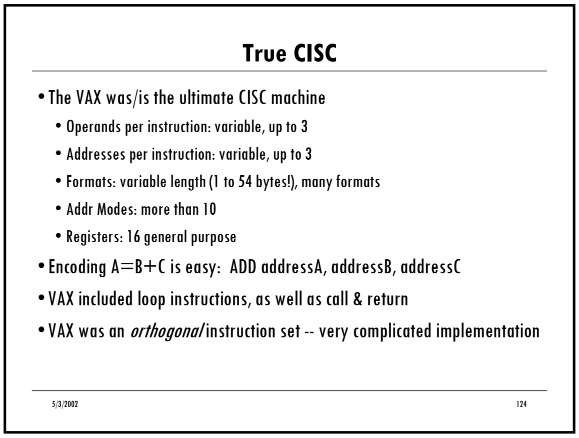
```
push     addressB
push     addressC
add
pop      addressA
```



CISC Machines

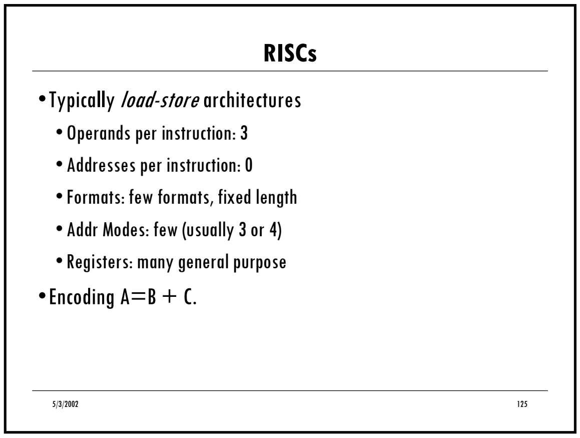
- Intel x86, Motorola 680x0 are examples
- They are *register-memory* architectures (some operands may be memory addresses)
  - Operands per instruction: variable, up to 2
  - Addresses per instruction: 1
  - Formats: variable length (x86 is between 1 and 17 bytes), many formats
  - Addr Modes: x86 has at least 7, 68k has more
  - Registers: usually some special purpose and some general
- Encode A = B + C

```
load      r1, addressB
add       r1, addressC
store     r1, addressA
```



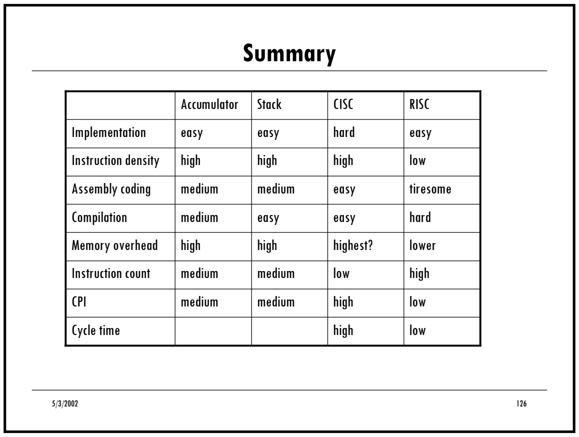
True CISC

- The VAX was/is the ultimate CISC machine
  - Operands per instruction: variable, up to 3
  - Addresses per instruction: variable, up to 3
  - Formats: variable length (1 to 54 bytes!), many formats
  - Addr Modes: more than 10
  - Registers: 16 general purpose
- Encoding A=B+C is easy: ADD addressA, addressB, addressC
- VAX included loop instructions, as well as call & return
- VAX was an *orthogonal* instruction set -- very complicated implementation



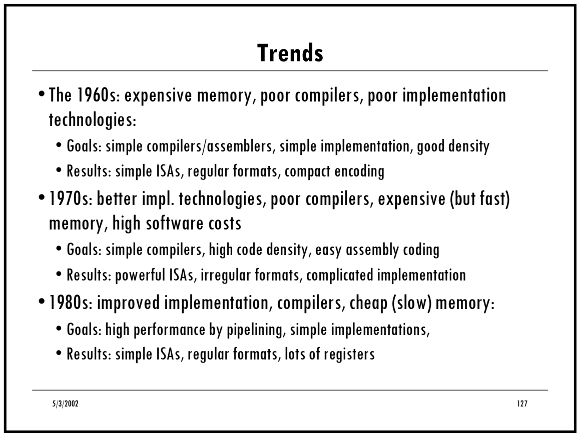
RISCs

- Typically *load-store* architectures
  - Operands per instruction: 3
  - Addresses per instruction: 0
  - Formats: few formats, fixed length
  - Addr Modes: few (usually 3 or 4)
  - Registers: many general purpose
- Encoding A=B + C.



Summary

	Accumulator	Stack	CISC	RISC
Implementation	easy	easy	hard	easy
Instruction density	high	high	high	low
Assembly coding	medium	medium	easy	tiresome
Compilation	medium	easy	easy	hard
Memory overhead	high	high	highest?	lower
Instruction count	medium	medium	low	high
CPI	medium	medium	high	low
Cycle time			high	low



Trends

- The 1960s: expensive memory, poor compilers, poor implementation technologies:
  - Goals: simple compilers/assemblers, simple implementation, good density
  - Results: simple ISAs, regular formats, compact encoding
- 1970s: better impl. technologies, poor compilers, expensive (but fast) memory, high software costs
  - Goals: simple compilers, high code density, easy assembly coding
  - Results: powerful ISAs, irregular formats, complicated implementation
- 1980s: improved implementation, compilers, cheap (slow) memory:
  - Goals: high performance by pipelining, simple implementations,
  - Results: simple ISAs, regular formats, lots of registers