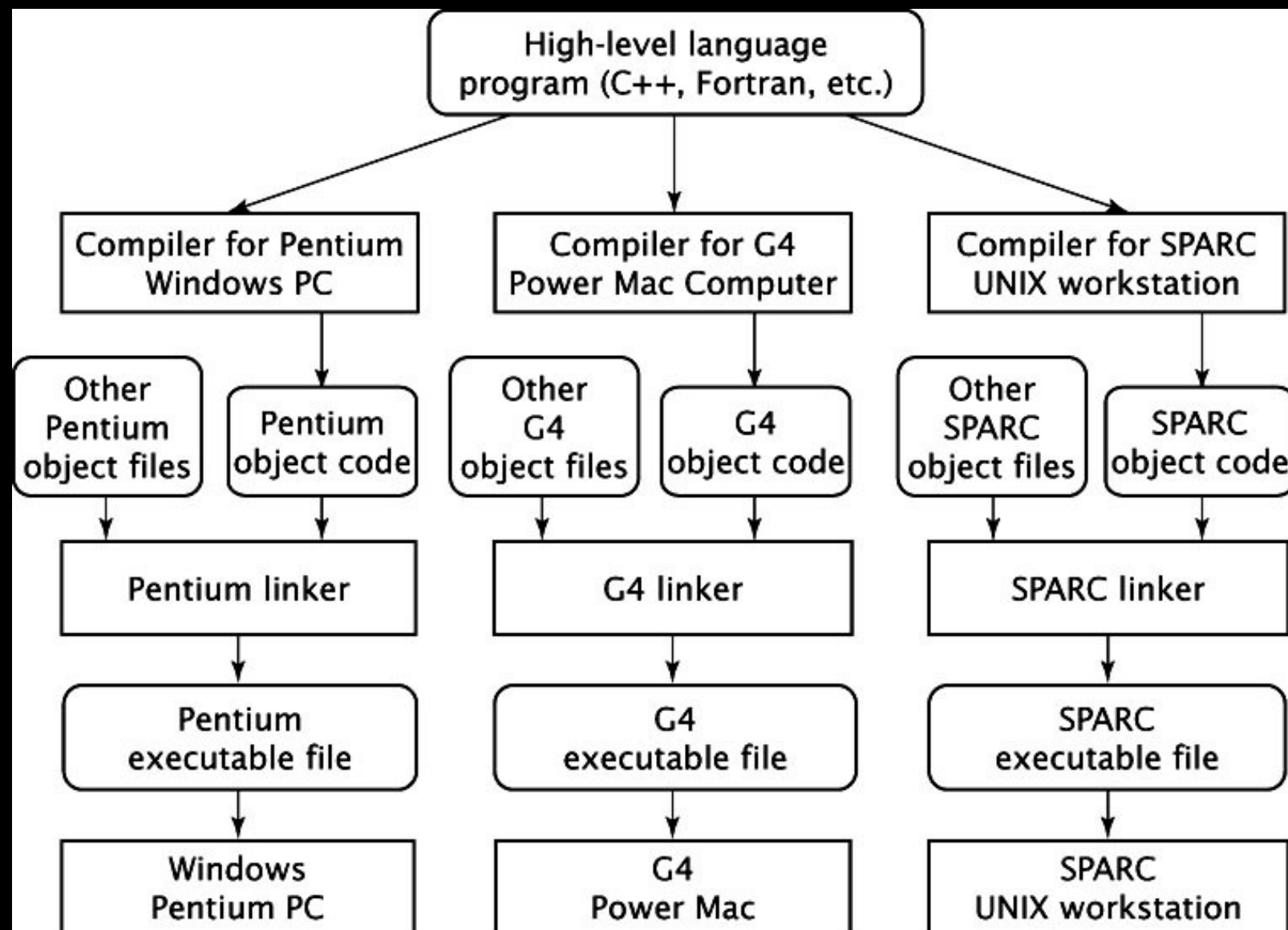


Instruction Set Architectures

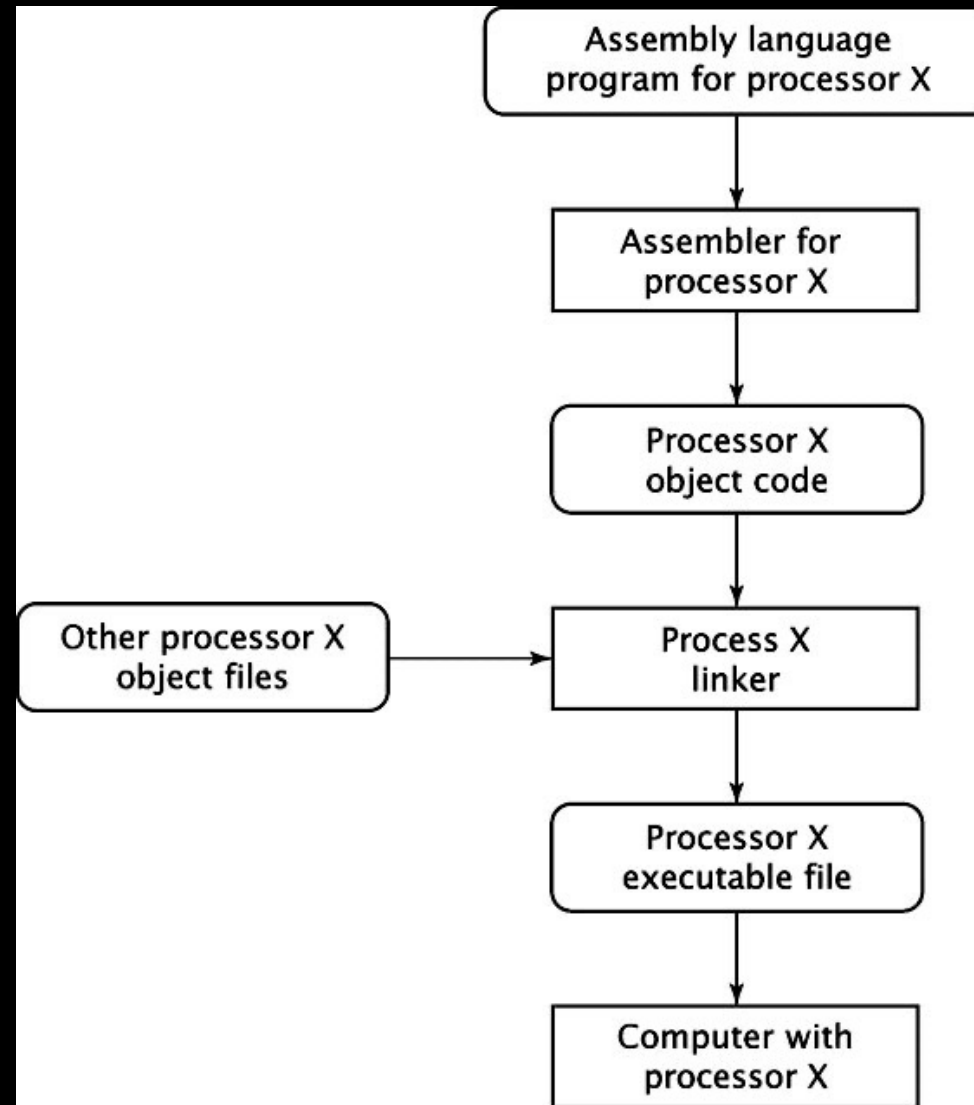
Programming Languages

- High level languages
- Assembly languages
- Machine languages

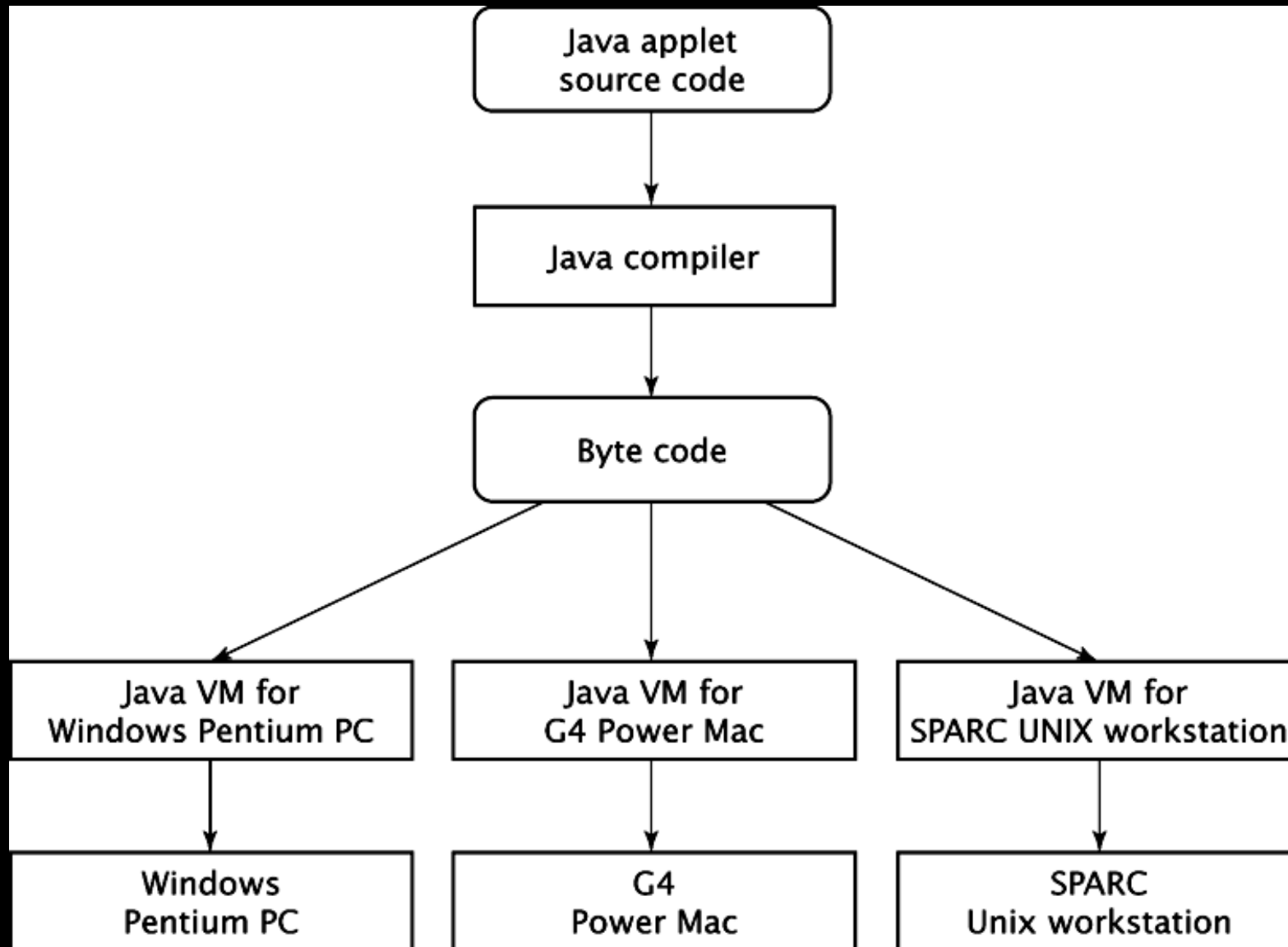
Compilation Process



Assembly Process



Java Applets



Assembly Language Attributes

- Instruction types
- Data types
- Addressing modes
- Instruction formats

Instruction Types

- Data transfer
- Data operation
- Program control

Data Types

- Numeric (integer, floating point)
- Boolean (true, false)
- Character (ASCII, Unicode)

Addressing modes - Direct

a) 0: LDAC 5
 ↙ instruction gets data from location 5
5: 10 → stores value in CPU

Addressing modes - Indirect

b) 0: LDAC @ 5

↙ instruction gets address from location 5

5: 10

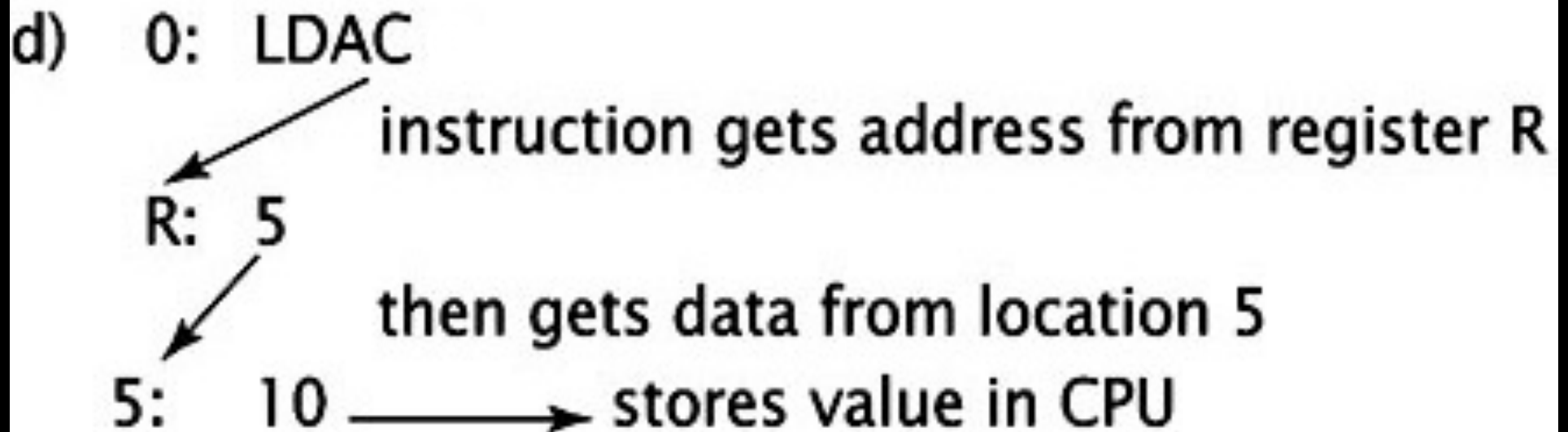
↙ then gets data from location 10

10: 20 → stores value in CPU

Addressing modes - Register Direct

c) 0: LDAC R
 ↙ instruction gets data from register R
R: 5 → stores value in CPU

Addressing modes - Register Indirect



Addressing modes - Immediate

e) 0: LDAC #5 → stores value from instruction in CPU

Addressing modes - Implicit

f) 0: LDAC (implicit)

instruction gets value from stack

stack → stores value in CPU

Addressing modes - Relative

g) 0: LDAC \$5
1: ↙ instruction adds address of next instruction (1) to
5: ↘ offset (5) to get address (6)
6: 12 → stores value in CPU

Addressing modes - Indexed

h) 0: LDAC 5(X)
 ↙ instruction gets value from index register
X: 10
 ↙ then adds contents of X (10) to offset (5) to get address (15)
15: 30 → stores value in CPU

Instruction Formats

- More operands = less instructions
- More operands = larger words

3-operand Instructions



2-operand Instructions

4 bits	2 bits	2 bits	
opcode	operand #1	operand #2	
			MOVE A,B (A=B) 1000 00 01
			ADD A,C (A=A+C) 1010 00 10

(b)

1-operand Instructions

4 bits		2 bits			
opcode		operand			
				LOAD <i>B</i>	(<i>Acc</i> = <i>B</i>) 0000 01
				ADD <i>C</i>	(<i>Acc</i> = <i>Acc</i> + <i>C</i>) 1010 10
				STORE <i>A</i>	(<i>A</i> = <i>Acc</i>) 0001 00

(c)

0-operand Instructions

4 bits

opcode

PUSH <i>B</i>	(Stack = <i>B</i>)	0101
PUSH <i>C</i>	(Stack = <i>C, B</i>)	0110
ADD	(Stack = <i>B + C</i>)	1010
POP <i>A</i>	(<i>A</i> = stack)	1100

(d)