LECTURE - 04

Announcements

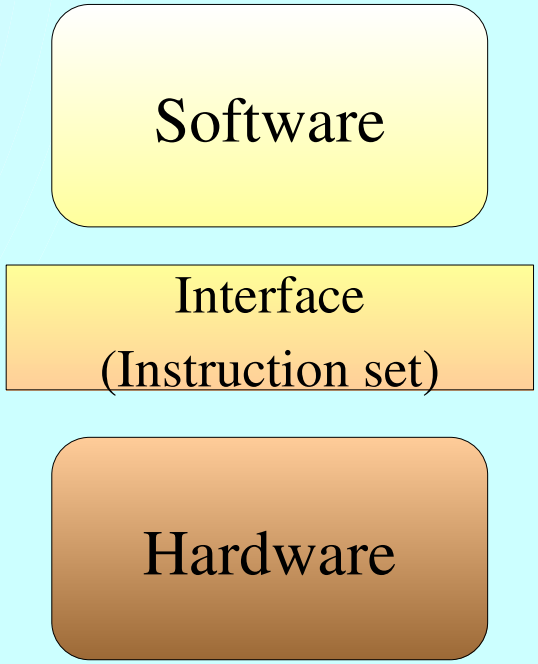
- Course web-page is up
  http://web.cse.iitk.ac.in/~cs422/index.html
- Lecture scribe notes:
  - HTML please
  - lec-notesXY-1.html or lec-notesXY-2.html
  - Images in directory “images/”
    - lecXY-1-anything.ext or lecXY-2-anything.ext
  - Please email to one of the TAs
- Extra classes?

Topics so far...

- Quantifying computer performance
- Amdahl's law
- Performance equation, CPI
- Effect of cache misses on CPI
- This week:
  - Instruction Set Architecture (ISA)
  - Pipelining: concept and issues

Instruction Set

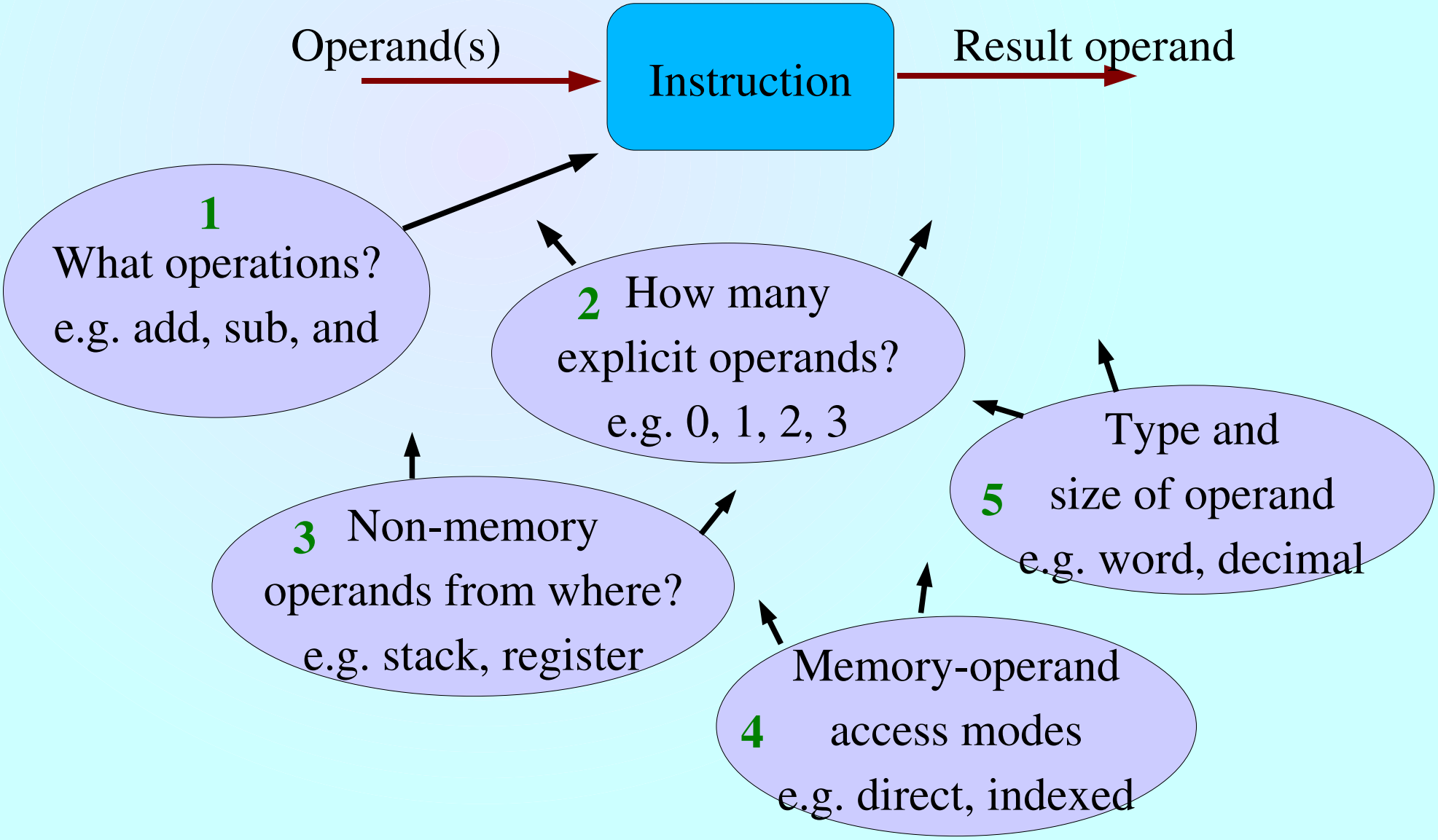
- Instruction set is the *interface* between hardware and software
- Interface design
  - Central part of any system design
  - Allows abstraction/independence
  - Challenges:
    - Should be easy to use by the layer above
    - Should allow efficient implementation by the layer below



Instruction Set Architecture (ISA)

- Main focus of early designs (1970s, 1980s)
- Mutual dependence between ISA design and:
  - Machine organization
    - Example: caches
  - Higher level languages and compilers (what instructions do they want?)
  - Operating systems
    - Example: atomic instructions, paging...

The Design Space



Other design choices: determining branch conditions, instruction encoding

Classes of ISAs

Stack	Accumulator	Register-memory	Register-register
Push A	Load A	Load R1, A	Load R1, A
Push B	Add B	Add R1, B	Load R2, B
Add	Store C	Store C, R1	Add R3, R1, R2
Pop C			Store C, R3

Memory-memory
Add C, A, B

- Those which use registers are also called ***General-Purpose Register (GPR)*** architectures
- Register-register also called ***load-store***

GPR Advantages

- Registers faster than memory
- Code density improves
- Easier for compiler to use
  - Hold variables
  - Expression evaluation
  - Passing arguments

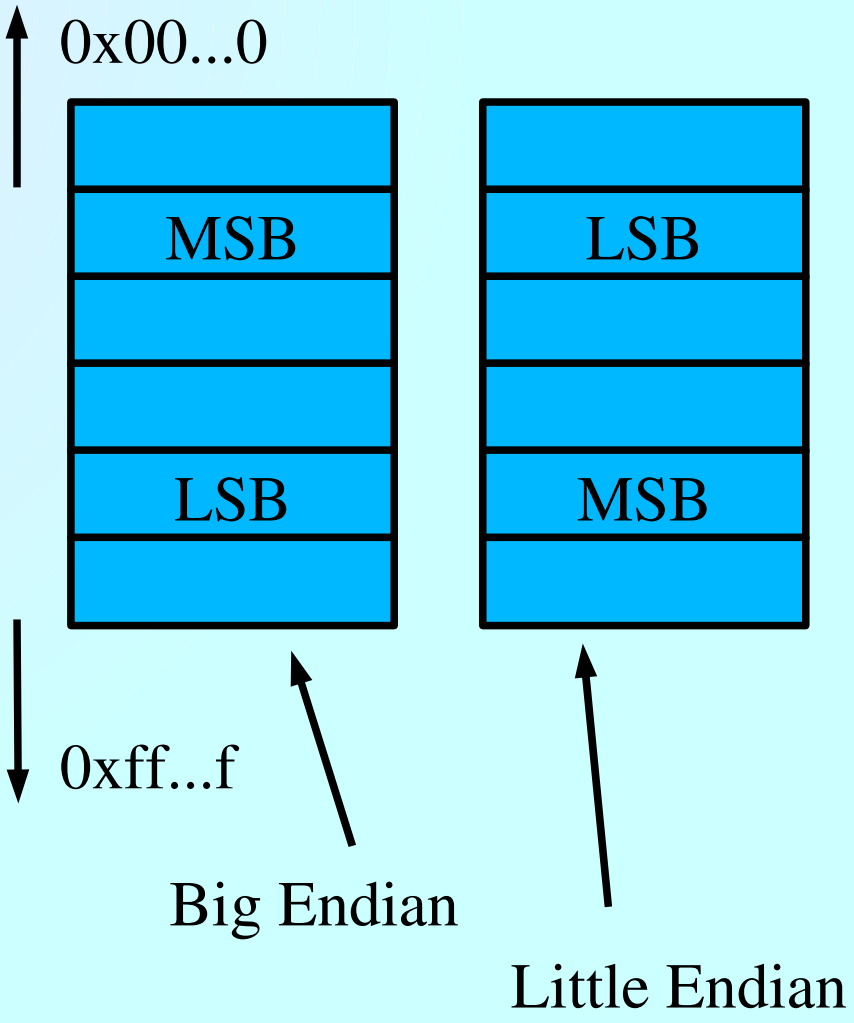
Spectrum of GPR Choices

- Choices based on
  - How many memory operands allowed
  - How many total operands

Number of memory addresses	Maximum number of operands allowed	Examples
0	3	SPARC, MIPS, PowerPC
1	2	80x86, Motorola
2	2	VAX

Memory Addressing

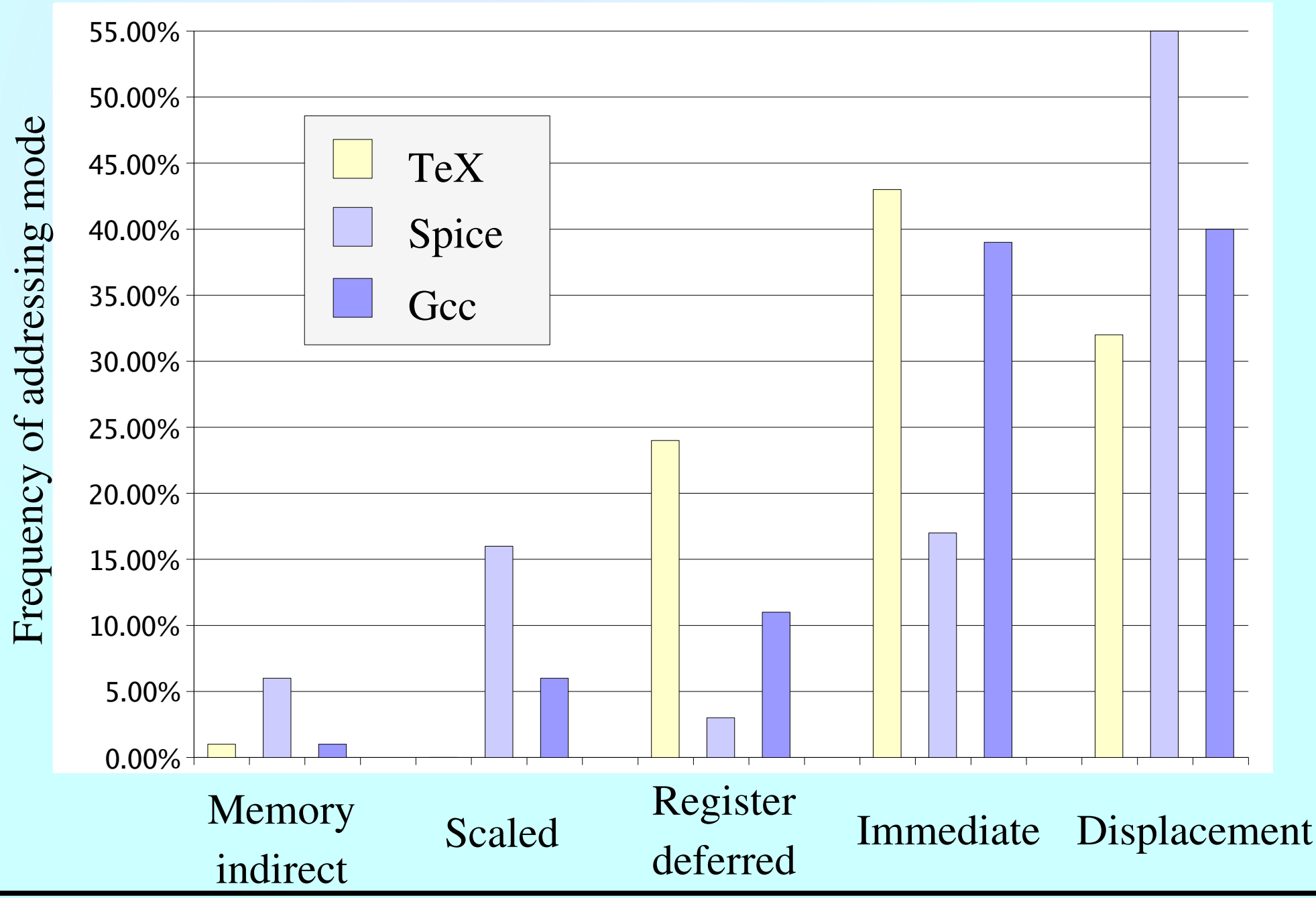
- Little-endian versus Big-endian
- Aligned versus non-aligned access of memory units > 1 byte
  - Misaligned ==> more memory cycles for access



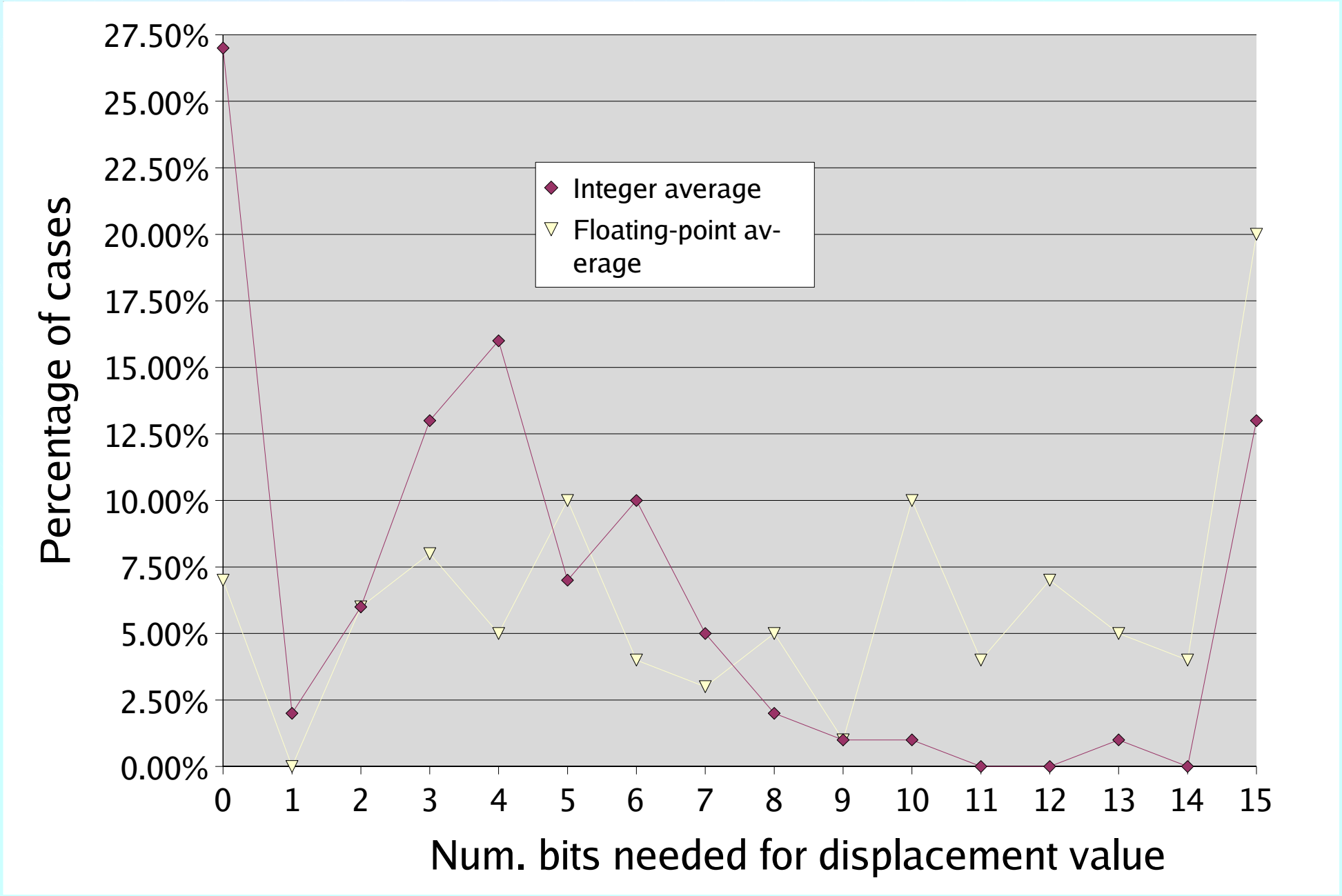
Addressing Modes

Addressing mode	Example	Meaning
Immediate	Add R4, #3	R4 <-- R4 + 3
Register	Add R4, R3	R4 <-- R4 + R3
Direct or absolute	Add R1, (1001)	R1 <-- R1 + M[1001]
Register deferred or indirect	Add R4, (R1)	R4 <-- R4 + M[R1]
Displacement	Add R4, 100(R1)	R4 <-- R4 + M[100+R1]
Indexed	Add R3, (R1+R2)	R3 <-- R3 + M[R1+R2]
Auto-increment	Add R1, (R2)+	R1 <-- R1 + M[R2]; R2 <-- R2 + d;
Auto-decrement	Add R1, –(R2)	R2 <-- R2 – d; R1 <-- R1 + M[R2]
Scaled	Add R1, 100(R2)[R3]	R1 <-- R1 + M[100+R2+R3*d]

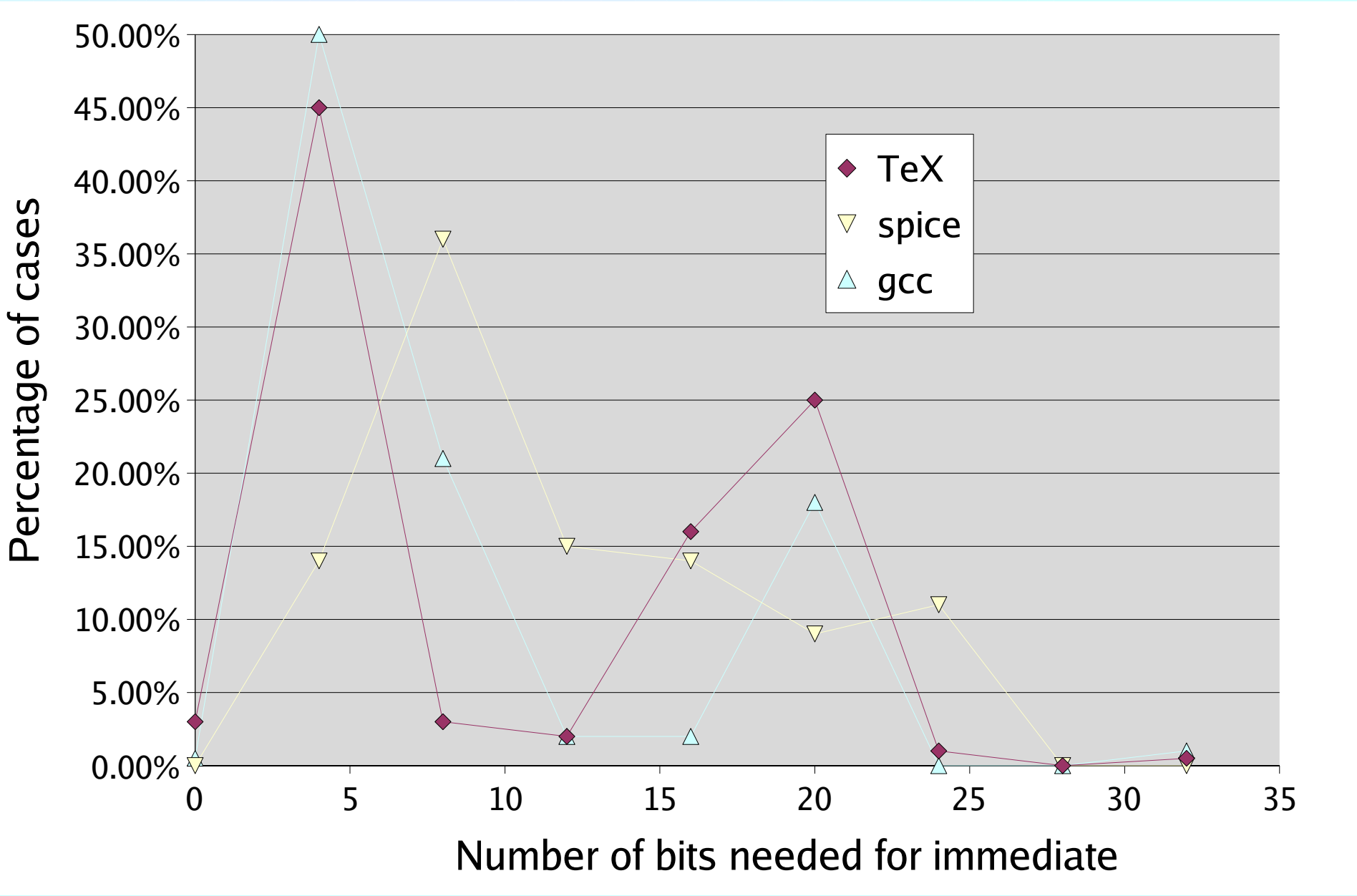
Usage of Addressing Modes



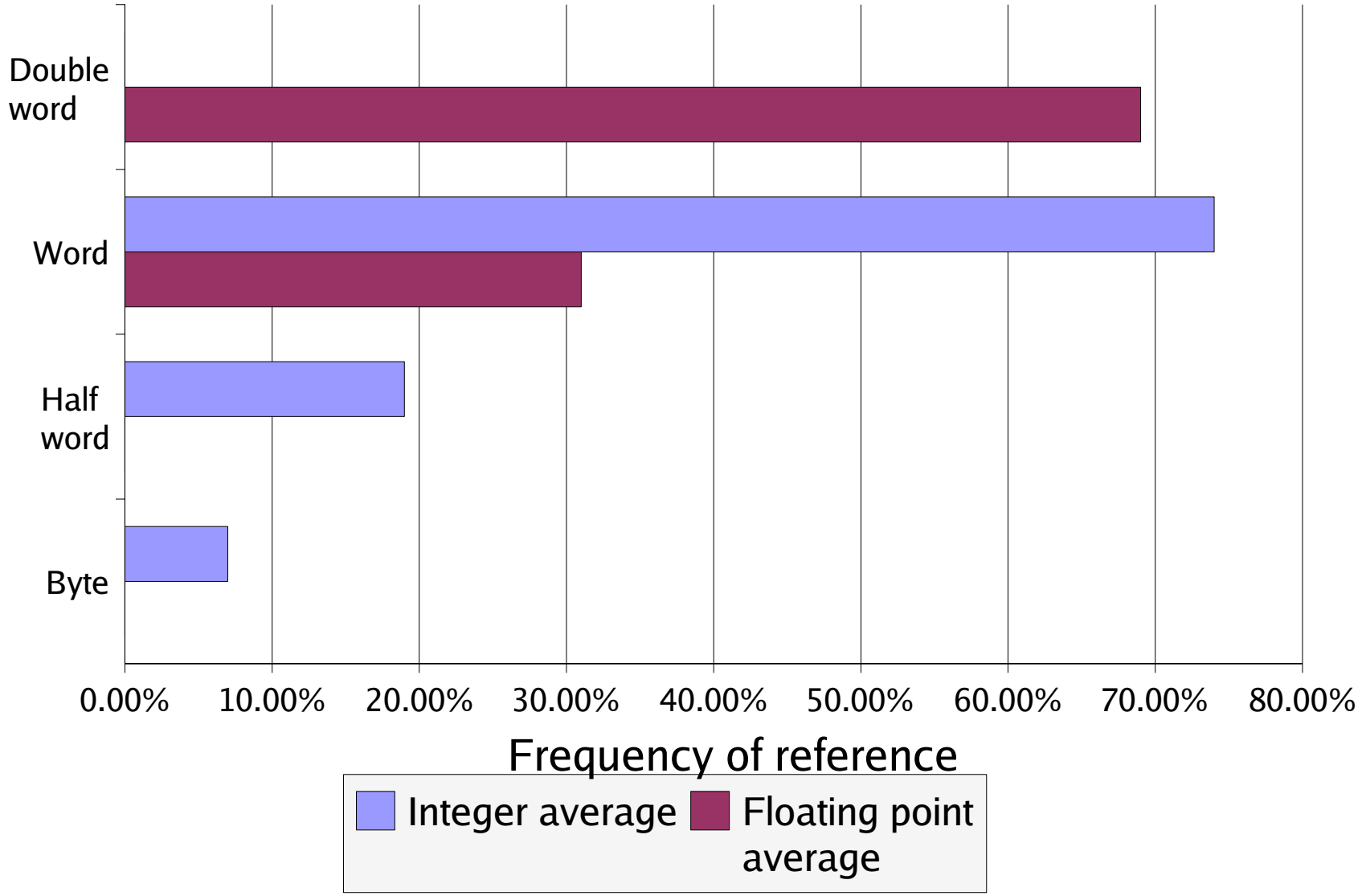
How many Bits for Displacement?



How many Bits for Immediate?



Type and Size of Operands



Summary so far

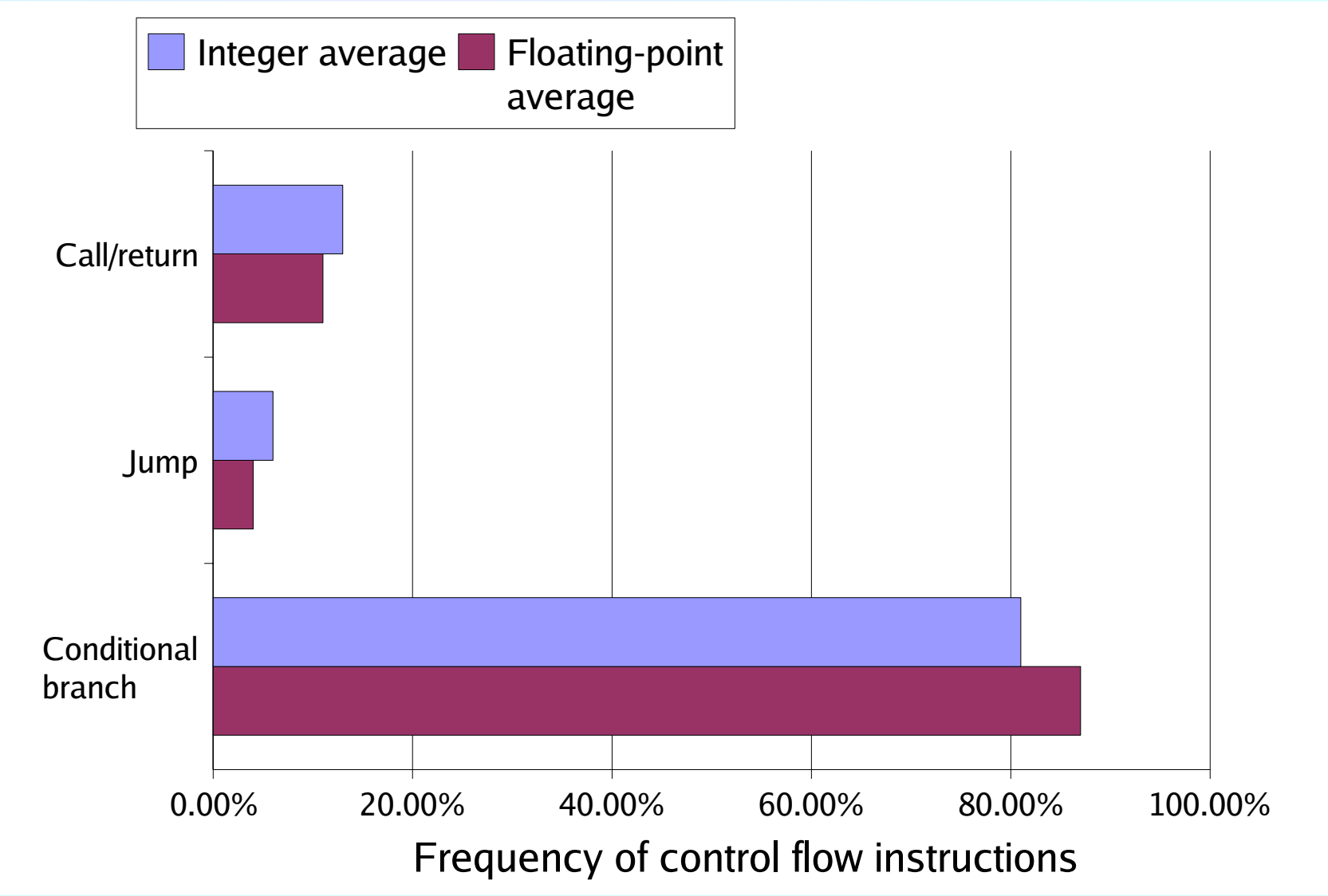
- GPR is better than stack/accumulator
- Immediate and displacement most used memory addressing modes
- Number of bits for displacement: 12-16 bits
- Number of bits for immediate: 8-16 bits
-
- Next: what operations in instruction set?

Deciding the Set of Operations

80x86 instruction	Integer average
Load	22.00%
Conditional branch	20.00%
Compare	16.00%
Store	12.00%
Add	8.00%
AND	6.00%
Sub	5.00%
Move reg-reg	4.00%
Call	1.00%

Simple instructions are used most!

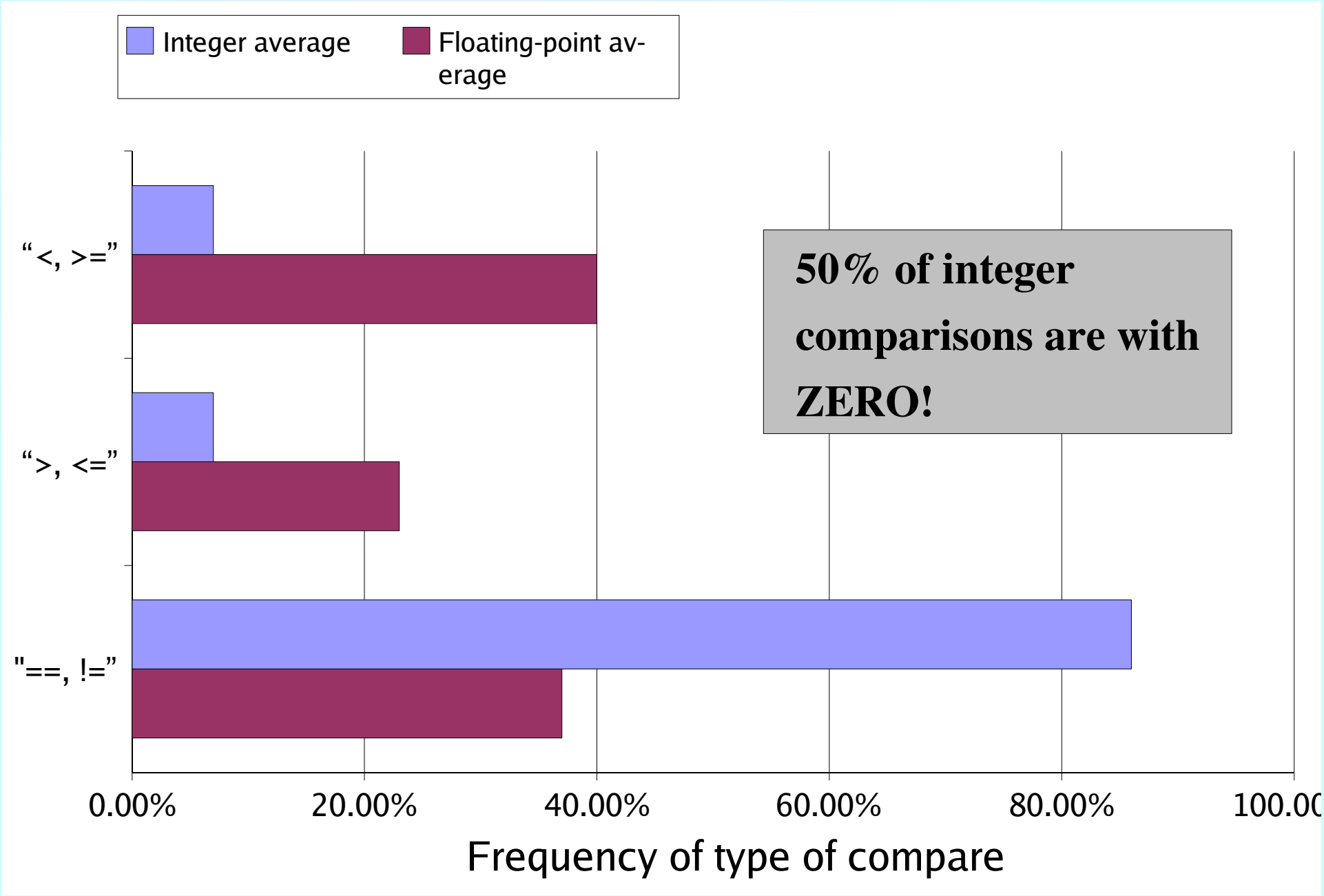
Instructions for Control Flow



Design Issues for Control Flow Instructions

- PC-relative addressing
  - Useful since most jumps/branches are nearby
  - Gives position independence (dynamic linking)
- Register indirect jumps
  - Useful for many programming language features
  - Case statements, virtual functions, dynamic libraries
- How many bits for PC displacement?
  - 8-10 bits are enough

What is the Nature of Compares?



Compare and Branch: Single Instruction or Two?

- Condition Code: set by ALU
  - Advantage: simple, may be free
  - Disadvantage: extra state across instructions
- Condition register: test any register with result of comparison
  - Advantage: simple
  - Disadvantage: uses up a register
- Compare and branch:
  - Advantage: lesser instructions
  - Disadvantage: too much work in an instruction

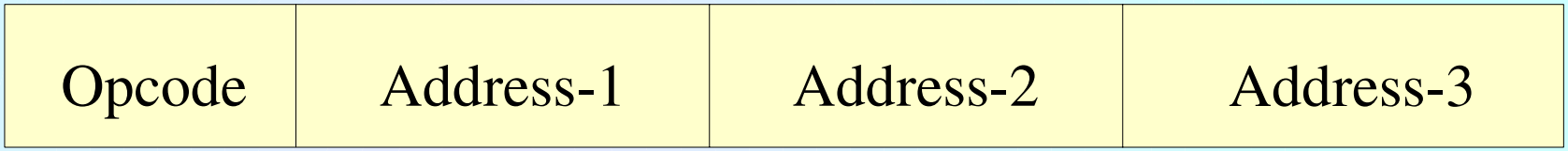
Managing Register State during Call/Return

- Caller save, or callee save?
  - Combination of the two is possible
- Beware of global variables in registers!

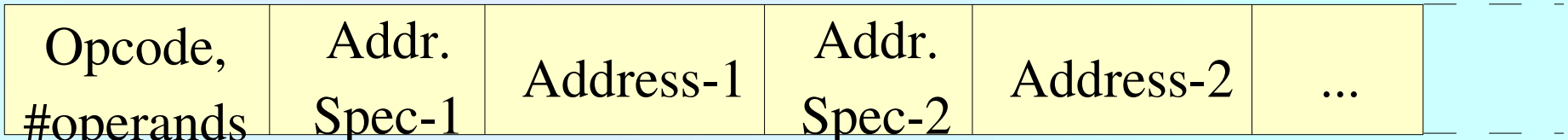
Instruction Encoding Issues

- Need to encode: operation, and addressing mode of each operand
  - *Opcode* is used for encoding operation
  - Simple set of addressing modes ==> can encode addressing mode also in opcode
  - Else, need *address specifier* per operand!
- Challenges in encoding:
  - Many registers and addressing modes
  - But, also minimize average instruction size
  - Encoding should be easy to handle in implementation (e.g. multiple of bytes)

Styles of Encoding



Fixed (e.g. DLX, MIPS, PowerPC)



Variable (e.g. VAX)

Fixed:

(+) ease of decoding

(--) more instructions

Variable:

(+) lesser number of instructions

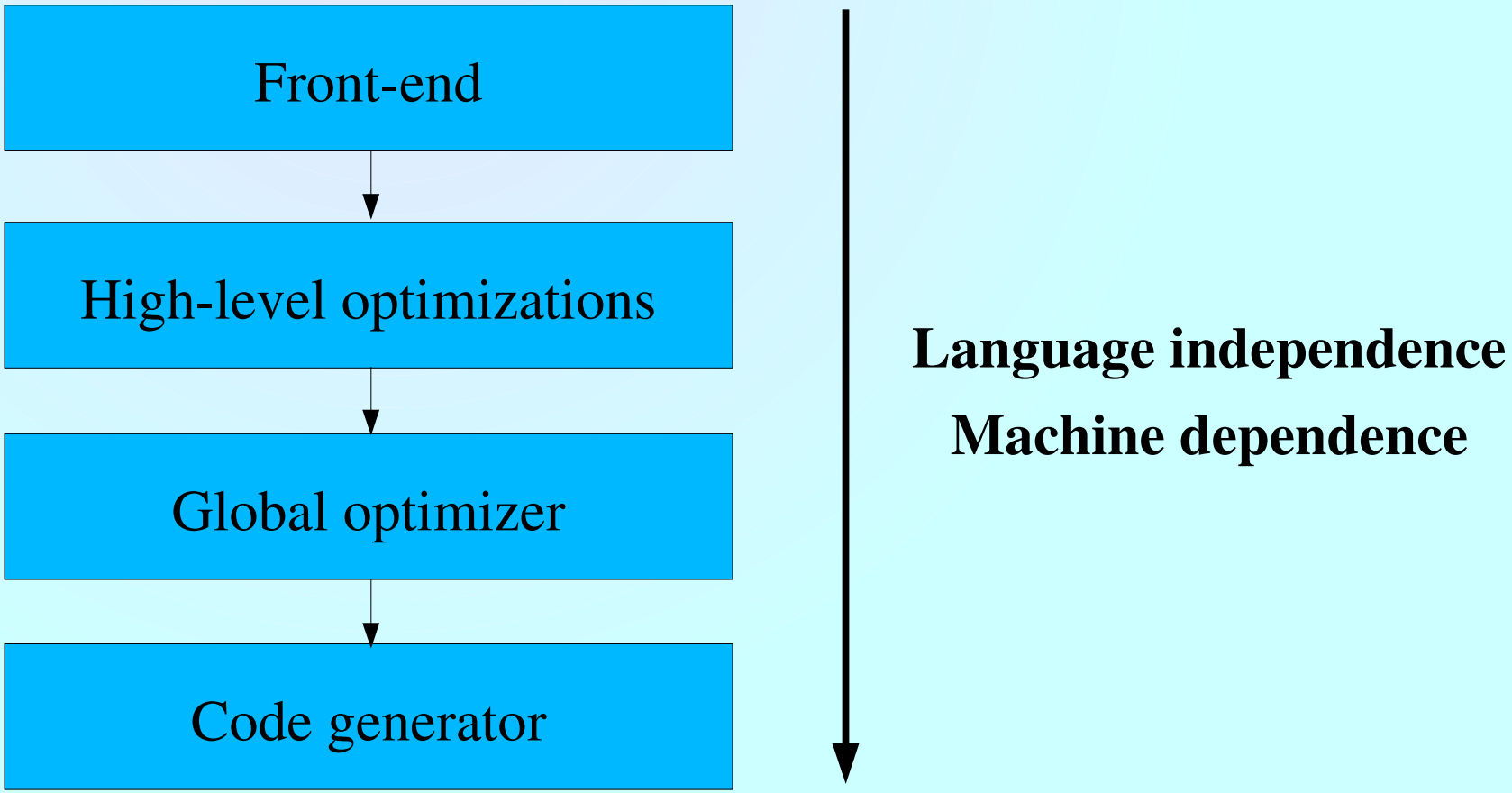
(--) variance in amount of work per instruction

Hybrid approach: reduce variability in size, but provide multiple encoding lengths

Examples: Intel 80x86

The Role of the Compiler

- Compilers are central to ISA design



ISA Design to Help the Compiler

- **Regularity:** operations, data-types, and addressing modes should be orthogonal; no special registers/operands for some instructions
- **Provide simple primitives:** do not optimize for a particular compiler of a particular language
- **Clear trade-offs among alternatives:** how to allocate registers, when to unroll a loop...

What lies ahead...

- The DLX architecture
- DLX: simple data-path
- DLX: pipelined data-path
- Pipelining hazards, and how to handle them