CMSC 611: Advanced Computer Architecture

Instruction Set Architecture

Register-Memory Arch

# memory addresses	Max. number of operands	Examples
0	3	SPARC, MIPS, PowerPC, ALPHA
1	2	Intel 60X86, Motorola 68000
2	2	VAX (also has 3 operands format)
3	3	VAX (also has 2 operands format)

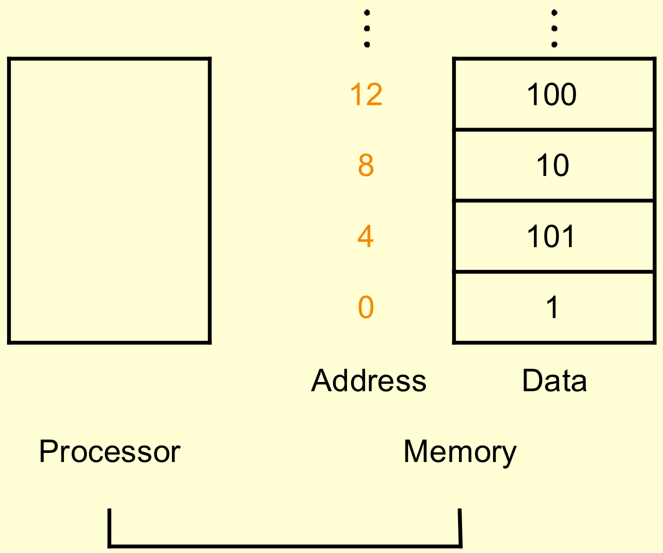
Effect of the number of memory operands:

Type	Advantages	Disadvantages
Reg-Reg (0,3)	- Fixed length instruction encoding - Simple code generation model - Similar execution time (pipeline)	- Higher instruction count - Some instructions are short leading to wasteful bit encoding
Reg-Mem (1,2)	- Direct access without loading - Easy instruction encoding	- Can restrict # register available for use - Clocks per instr. varies by operand type - Source operands are destroyed
Mem-Mem (3,3)	- No temporary register usage - Compact code	- Less potential for compiler optimization - Can create memory access bottleneck

Memory Addressing

- The address of a word matches the byte address of one of its 4 bytes
- The addresses of sequential words differ by 4 (word size in byte)
- Words' addresses are multiple of 4 (alignment restriction)
  - Misalignment (if allowed) complicates memory access and causes programs to run slower

Object addressed	Aligned at byte offsets	Misaligned at byte offsets
Byte	1,2,3,4,5,6,7	Never
Half word	0,2,4,6	1,3,5,7
Word	0,4	1,2,3,5,6,7
Double word	0	1,2,3,4,5,6,7



Byte Order

- Given N bytes, which is the most significant, which is the least significant?
  - "Big Endian"
    - Leftmost / most significant byte = word address
    - Intel (among others)
  - "Little Endian"
    - Rightmost / least significant byte = word address
    - Motorola, TCP/IP (among others)
- Byte ordering can be as problem when exchanging data among different machines
- Can also affect array index calculation or any other operation that treat the same data a both byte and word.

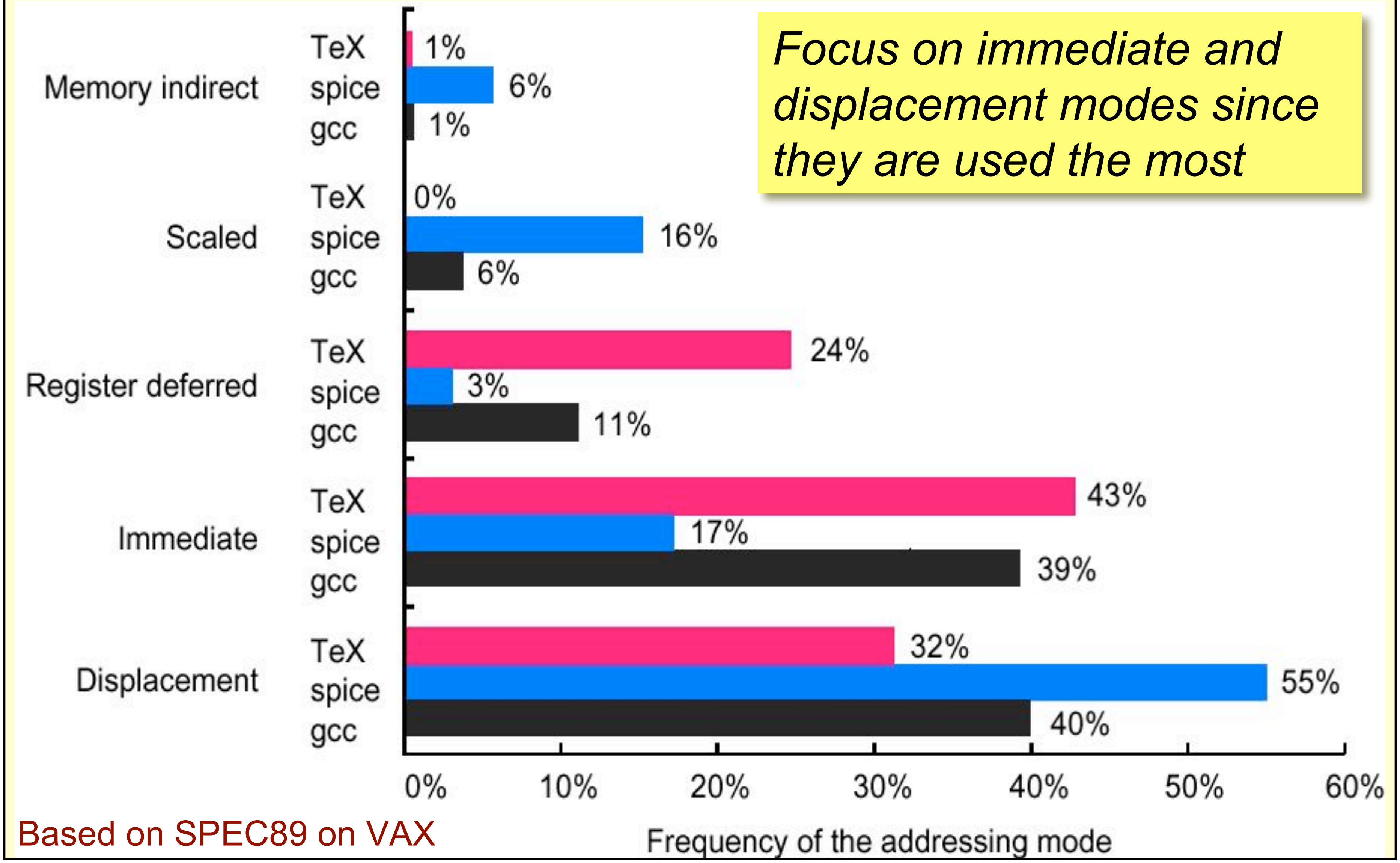
Addressing Modes

- How to specify the location of an operand (effective address)
- Addressing modes have the ability to:
  - Significantly reduce instruction counts
  - Increase the average CPI
  - Increase the complexity of building a machine
- VAX machine is used for benchmark data since it supports wide range of memory addressing modes
- Can classify based on:
  - source of the data (register, immediate or memory)
  - the address calculation (direct, indirect, indexed)

Example of Addressing Modes

Mode	Example	Meaning	When used
Register	ADD R4, R3	*Regs[R4] = Regs[R4] + Regs[R3]*	When a value is in a register
Immediate	ADD R4, #3	*Regs[R4] = Regs[R4] + 3*	For constants
Register indirect	ADD R4, (R1)	*Regs[R4] = Regs[R4] + Mem[Regs[R1]]*	Accessing using a pointer or a computed address
Direct or absolute	ADD R4, (1001)	*Regs[R4] = Regs[R4] + Mem[1001]*	Sometimes useful for accessing static data; address constant may need to be large
Displacement	ADD R4, 100 (R1)	*Regs[R4] = Regs[R4] + Mem[100 + Regs[R1]]*	Accessing local variables
Indexed	ADD R4, (R1 + R2)	*Regs[R4] = Regs[R4] + Mem[Regs[R1] + Regs[R2]]*	Sometimes useful in array addressing: R1 = base of the array: R2 = index amount
Autoincrement	ADD R4, (R2) +	*Regs[R4] = Regs[R4] + Mem[Regs[R2]]* *Regs[R2] = Regs[R2] + d*	Useful for stepping through arrays within a loop. R2 points to start of the array; each reference increments R2 by d.
Auto decrement	ADD R4, -(R2)	*Regs[R2] = Regs[R2] – d* *Regs[R4] = Regs[R4] + Mem[Regs[R2]]*	Same use as autoincrement. Autodecrement/increment can also act as push/pop to implement a stack
Scaled	ADD R4, 100 (R2) [R3]	*Regs[R4] = Regs[R4] + Mem[100 + Regs[R2] + Regs[R3] * d]*	Used to index arrays.

Addressing Mode Use

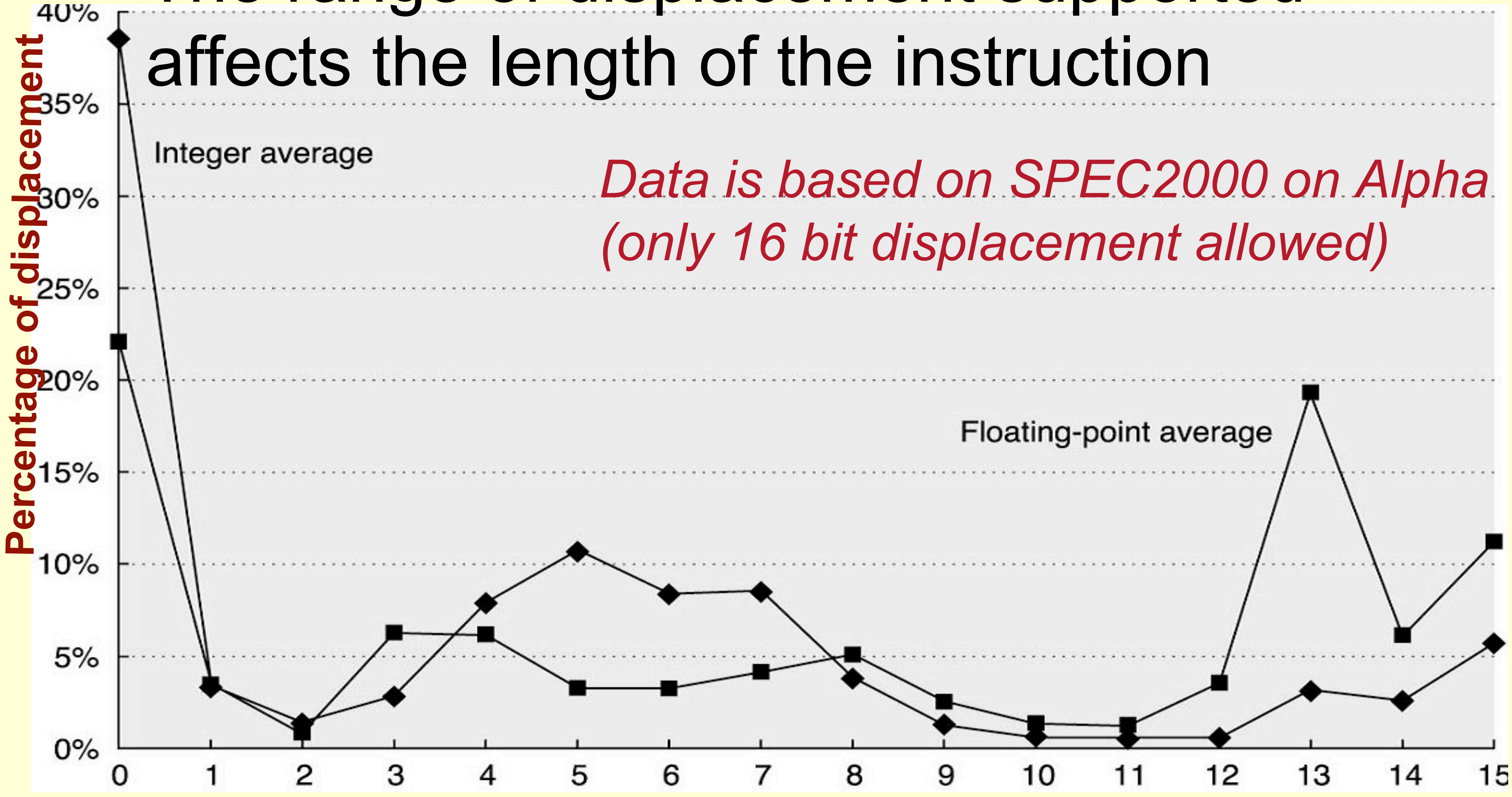


Focus on immediate and displacement modes since they are used the most

Based on SPEC89 on VAX

Displacement Addressing Modes

- The range of displacement supported affects the length of the instruction

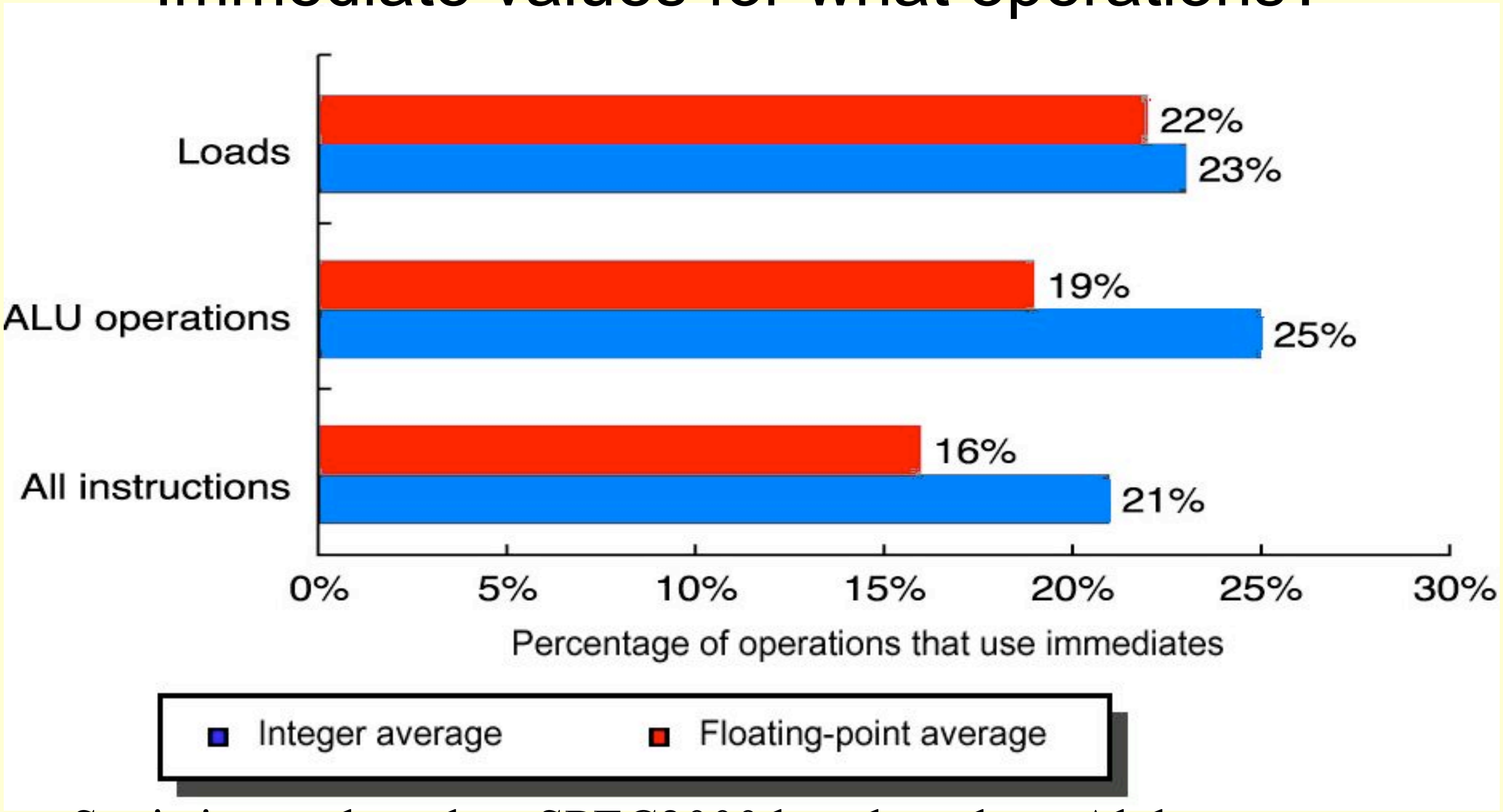


Data is based on SPEC2000 on Alpha (only 16 bit displacement allowed)

Number of bits needed for a displacement value in SPEC2000 benchmark

Immediate Addressing Modes

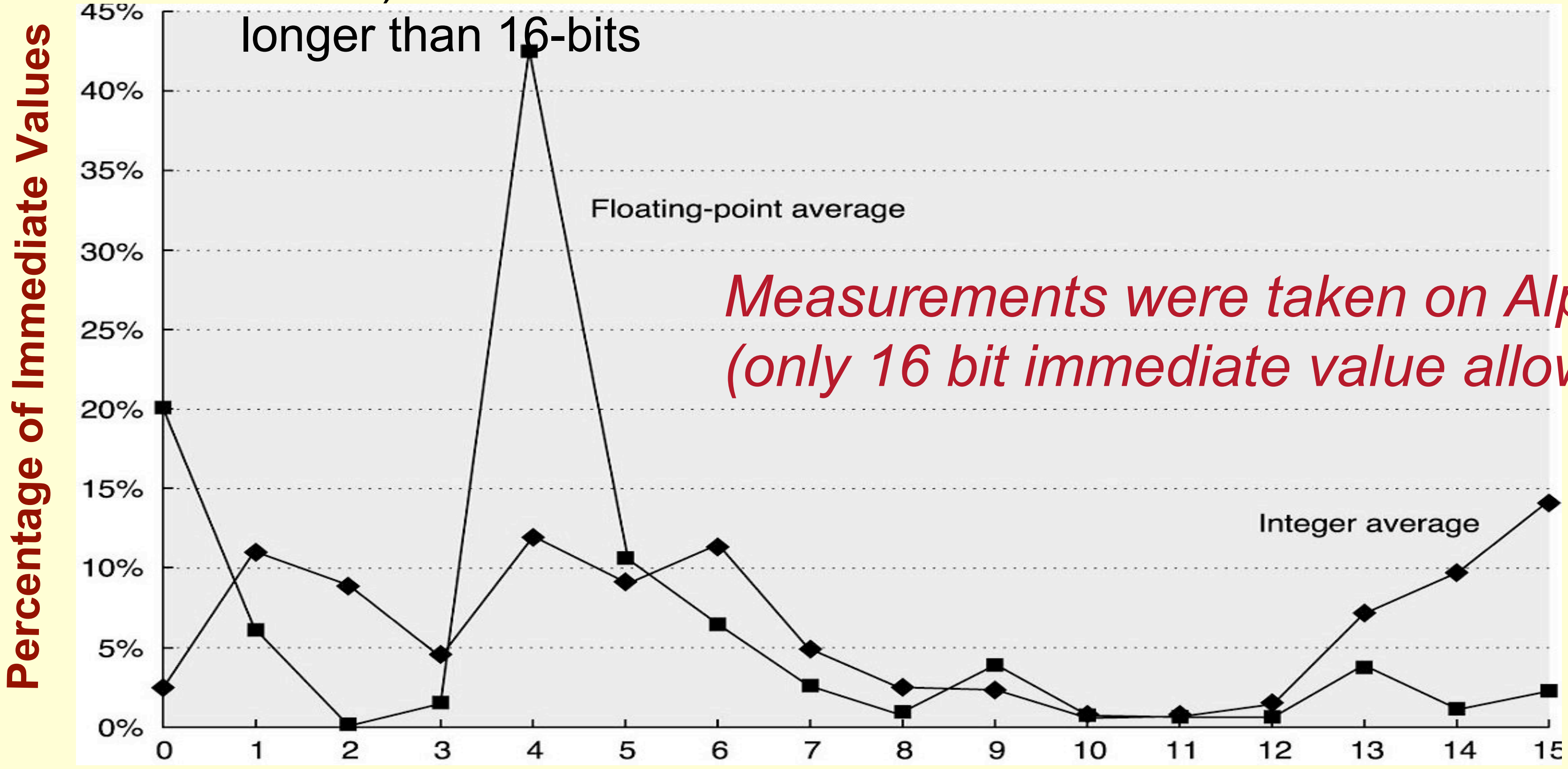
- Immediate values for what operations?



Statistics are based on SPEC2000 benchmark on Alpha

Distribution of Immediate Values

- Range affects instruction length
  - Similar measurements on the VAX (with 32-bit immediate values) showed that 20-25% of immediate values were longer than 16-bits



Measurements were taken on Alpha (only 16 bit immediate value allowed)

Number of bits needed for a immediate values in SPEC2000 benchmark

Addressing Mode for Signal Processing

- DSP offers special addressing modes to better serve popular algorithms
- Special features requires either hand coding or a compiler that uses such features

Addressing Mode for Signal Processing

- Modulo addressing:
  - Since DSP deals with continuous data streams, circular buffers common
  - Circular or modulo addressing: automatic increment and decrement / reset pointer at end of buffer
- Reverse addressing:
  - Address is the reverse order of the current address
  - Expedites access / otherwise require a number of logical instructions or extra memory accesses

<u>**Fast Fourier Transform**</u>

0 (000_2) ➔ 0 (000_2)

1 (001_2) ➔ 4 (100_2)

2 (010_2) ➔ 2 (010_2)

3 (011_2) ➔ 6 (110_2)

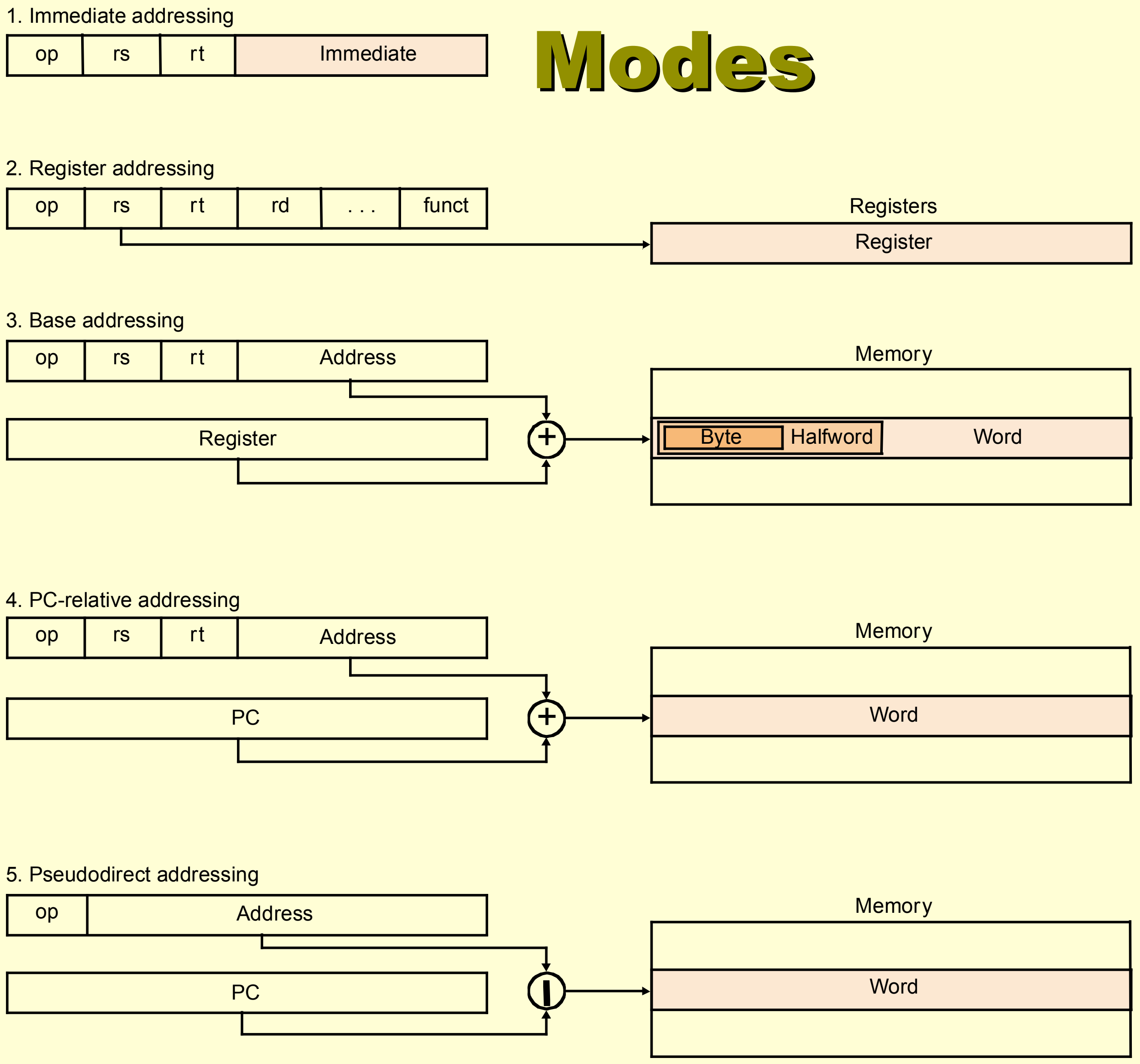
4 (100_2) ➔ 1 (001_2)

5 (101_2) ➔ 5 (101_2)

6 (110_2) ➔ 3 (011_2)

7 (111_2) ➔ 7 (111_2)

Summary of MIPS Addressing Modes



Operations of the Computer Hardware

“There must certainly be instructions for performing the fundamental arithmetic operations.”

Burkes, Goldstine and Von Neumann, 1947

MIPS assembler allows only one instruction/line and ignore comments following # until end of line

Example:

Translation of a segment of a C program to MIPS assembly instructions:

C:

```
f = (g + h) - (i + j)
```

(pseudo)MIPS:

```
add     t0, g, h        # temp. variable t0 contains "g + h"
add     t1, i, j        # temp. variable t1 contains "i + j"
sub     f, t0, t1       # f = t0 - t1 = (g + h) - (i + j)
```

Operations in the Instruction Set

Operator type	Examples
Arithmetic and logical	Integer arithmetic and logical operations: add, and, subtract , or
Data Transfer	Loads-stores (move instructions on machines with memory addressing)
Control	Branch, jump, procedure call and return, trap
System	Operating system call, Virtual memory management instructions
Floating point	Floating point instructions: add, multiply
Decimal	Decimal add, decimal multiply, decimal to character conversion
String	String move, string compare, string search
Graphics	Pixel operations, compression/decompression operations

- Arithmetic, logical, data transfer and control are almost standard categories for all machines
- System instructions are required for multi-programming environment although support for system functions varies
- Others can be primitives (e.g. decimal and string on IBM 360 and VAX), provided by a co-processor, or synthesized by compiler.

Operations for Media & Signal Process.

- Partitioned Add:
  - Partition a single register into multiple data elements (e.g. 4 16-bit words in 1 64-bit register)
  - Perform the same operation independently on each
  - Increases ALU throughput for multimedia applications
- Paired single operations
  - Perform multiple independent narrow operations on one wide ALU (e.g. 2 32-bit float ops)
  - Handy in dealing with vertices and coordinates
- Multiply and accumulate
  - Very handy for calculating dot products of vectors (signal processing) and matrix multiplication

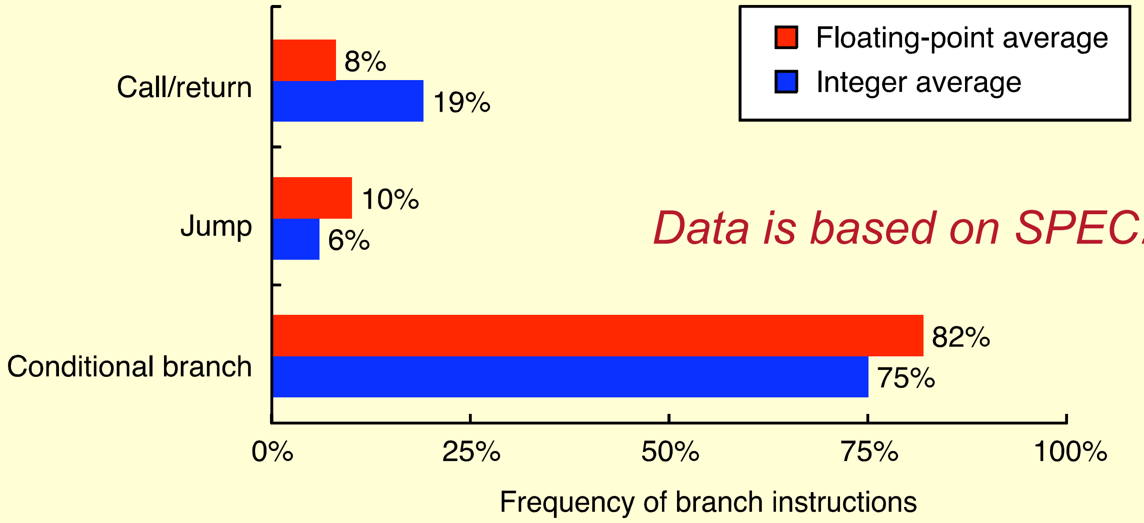
Frequency of Operations Usage

- The most widely executed instructions are the simple operations of an instruction set
- Average usage in SPECint92 on Intel 80x86:

Rank	80x86 Instruction	Integer Average (% total executed)
1	Load	22%
2	Conditional branch	20%
3	Compare	16%
4	Store	12%
5	Add	8%
6	And	6%
7	Sub	5%
8	Move register-register	4%
9	Call	1%
10	Return	1%
Total		96%

Make the common case fast by focusing on these operations

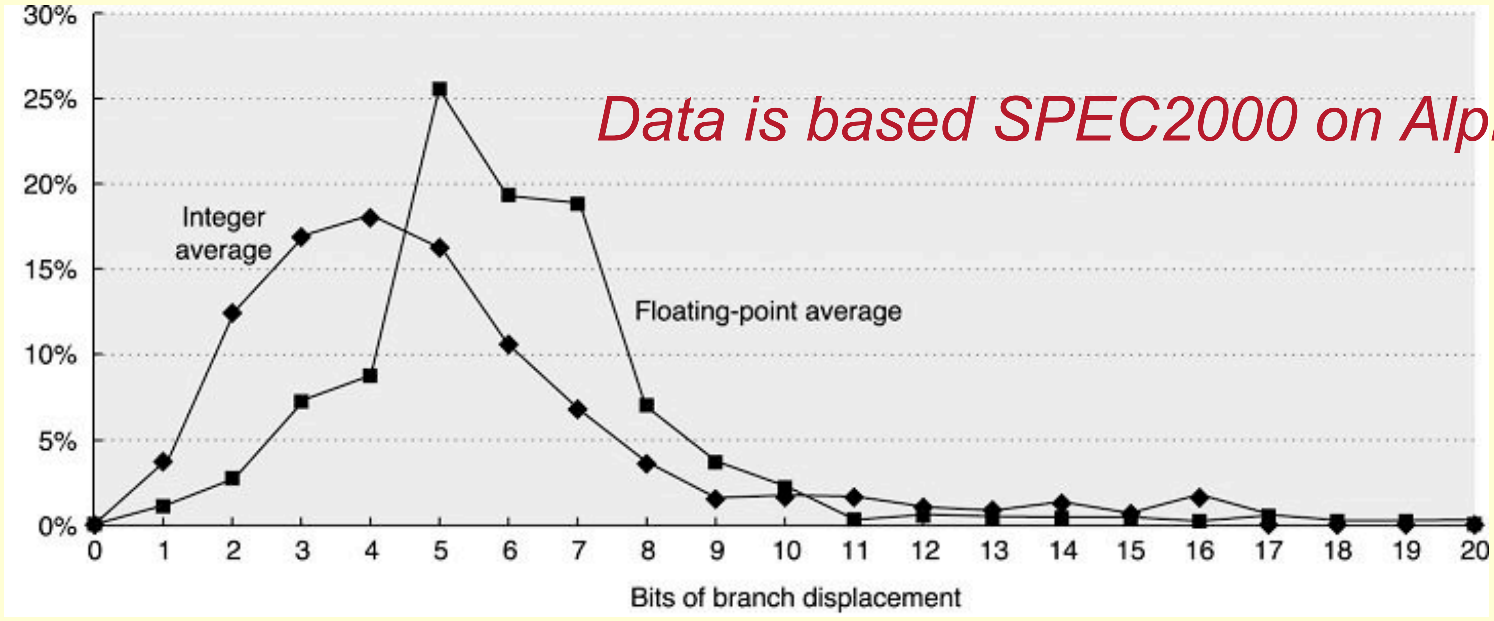
Control Flow Instructions



Data is based on SPEC2000 on Alpha

- Jump: unconditional change in the control flow
- Branch: conditional change in the control flow
- Procedure calls and returns

Destination Address Definition



Data is based SPEC2000 on Alpha

- PC-relative addressing
  - Good for short position-independent forward & backward jumps
- Register indirect addressing
  - Good for dynamic libraries, virtual functions & packed case statements