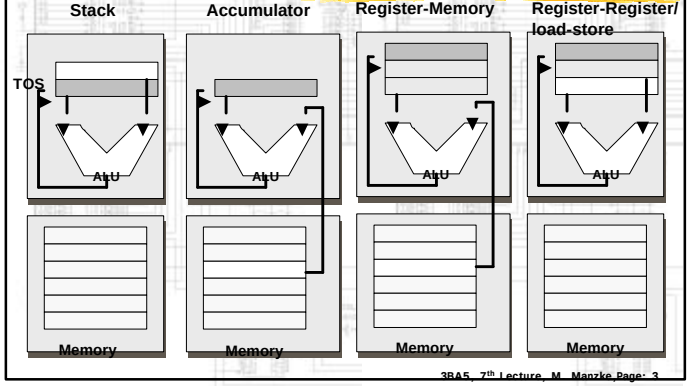
Instruction Set Architecture (ISA)

- Concerned with the portion that is visible to the programmer and compiler developer.
- We look at various design alternatives.

Application domains

- Desktop machines require:
  - Integer and floating-point performance
  - Program size and power consumption is less important
- Servers (database, file server and web applications) require:
  - Integer and character strings
  - Floating operation are less important but supported
- Embedded application require:
  - Small memory and less instructions (floating point) to save cost and power.
- DSP / Media processors support continuous streams of data at real-time (many general purpose processors have media instructions)

ISA Taxonomy



C = A + B

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

Register-Register (0,3)

- Number of memory addresses:
  - 0
- Max number of operands:
  - 3
- Example:
  - Alpha, ARM, MIPS, PowerPC, SPARC, SuperH, Trimedia
- Advantages:
  - Fixed-length instruction encoding. Simple code generation. Instruction take similar number of clock cycles to execute.
- Disadvantages:
  - More instructions and lower instruction density leads to larger programs.

Register-Memory (1,2)

- Number of memory addresses:
  - 1
- Max number of operands:
  - 2
- Example:
  - IBM360/370, Intel 80x86, Motorola 68000, TI TMS320C54x
- Advantages:
  - Data can be accessed without a separate load instruction first. Instruction format tends to be easy to encode.
- Disadvantages:
  - Operands are not equivalent since a source operand in binary operations can be destroyed. Clocks per instruction vary by operand location.

Memory-Memory (2,2)/(3,3)

- Number of memory addresses:
  - 2/3
- Max number of operands:
  - 2/3
- Example:
  - VAX (Has two-operand and three operand formats)
- Advantages:
  - Most compact. Doesn't waste registers for temporaries.
- Disadvantages:
  - Large variations in the instruction size. Memory access create memory bottleneck.

Memory Addressing

- The IS must specify how memory addresses are defined and interpreted
- Interpreting Memory Addresses:
  - Accessing an object depends on address on length
  - Byte ordering in a larger object(Gulliver's Travels)
    - Little Endean (puts the smallest address at the least-significant position)
    - Big Endean (puts the smallest address at the most-significant position)
  - Aligned address

Addressing Modes

- Specify the address of an object to be accessed
- Addressing modes specify:
  - Constants
  - Registers
  - Memory locations
- Memory address defined by the addressing mode is called the:
  - Effective address
- Addressing modes can significantly reduce instruction counts
- They add complexity and may increase the average CPI (clock cycles per instruction)

Addressing Modes[1]

Addressing mode	Example instruction	Register transfer	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.
Displacement	Add R4,100(R1)	Regs[R4]<- Regs[R4] + Mem[100+Regs[R1]]	Accessing local variables and simulates register indirect/direct addressing
Register indirect	Add R4,(R1)	Regs[R4]<- Regs[R4] + Mem[Regs[R1]]	Accessing using a pointer
Index	Add R3,(R1+R2)	Regs[R3]<- Regs[R3] + Mem[Regs[R1]+Regs[R2]]	Useful in array addressing

Addressing Modes[2]

Addressing mode	Example instruction	Register transfer	When used
Direct or absolute	Add R1,(1001)	Regs[R1]<- Mem[1001]	Useful to access static data.
Memory indirect	Add R1,@(R3)	Regs[R1]<- Mem[MEM[Regs[R3]]]	If R3 is the address of a pointer p, then mode yields *p.
Autoincrement	Add R1,(R2)+	Regs[R1]<- Mem[Mem[Regs[R2]] Regs[R2]<- Regs[R2]+d	Stepping through arrays.
Autodecrement	Add R1,-(R2)	Regs[R2]<- Regs[R2]-d Regs[R1]<- Mem[Mem[Regs[R2]]	Stepping through arrays.
Scaled	Add R1,100(R2)[R3]	Regs[R1]<-Regs[R1] + Mem[100+Regs[R2]+ Regs[R3]*d]	Used to index arrays.

Addressing Mode Usage Patterns

- The VAX architecture has an rich set of addressing modes
- The graph on the following slide show that Immediate and Displacement modes dominate.
- Also register modes are not taken into account.
  - They account for 50% of the operand references
- The compiler determines the instructions
- A SPEC89 program was executed on the VAX to generate the following graph.