Memory Addressing

- Least addressable unit is byte (8 bits)
- Also accessible: half-word (16 bits), word (32 bits), double word (64 bits)
- How are longer objects stored? 
  - Big endian: AB 10 23 45
  - Least significant byte first
- Little endian: 45 23 10 AB
  - Most significant byte first
- Accesses to objects larger than byte are aligned
- Size \( s \)
  - \( A \) mod \( s = 0 \)

Addressing Modes

- How to specify location of an object:
  - Memory location
  - Register
  - Constant

- Register:
  - operand is in register
  - Add R4, R3
  - \( \text{Regs}[R4] \leftarrow \text{Regs}[R4]+\text{Regs}[R3] \)

- Immediate:
  - operand is constant
  - Add R4, #3
  - \( \text{Regs}[R4] \leftarrow \text{Regs}[R4]+3 \)

- Register indirect:
  - operand is in memory location specified by a register
  - Add R4, (R1)
  - \( \text{Regs}[R4] \leftarrow \text{Regs}[R4]+\text{Mem}[\text{Regs}[R1]] \)

- Displacement:
  - operand is in memory location specified by a register and some displacement
  - Add R4, 100(R1)
  - \( \text{Regs}[R4] \leftarrow \text{Regs}[R4]+\text{Mem}[\text{Regs}[R1]+100] \)

- Indexed:
  - operand is in a memory location specified by a sum of the two register contents
  - Add R3, (R1+R2)
  - \( \text{Regs}[R3] \leftarrow \text{Regs}[R3]+\text{Mem}[\text{Regs}[R1]+\text{Regs}[R2]] \)

- Direct or absolute:
  - operand is in a memory location specified by a constant
  - Add R1, (1001)
  - \( \text{Regs}[R1] \leftarrow \text{Regs}[R3]+\text{Mem}[1001] \)
Addressing Modes

- **Memory indirect**: operand is in a memory location which is specified by another memory location, stored in a register
  Add R1, @ (R3)
  Regs[R1] ← Regs[R1] + Mem[Mem[Regs[R3]]]

- **Autoincrement**: operand is in a memory location specified by a register; register value is postincremented by operand size
  Add R1, (R2)+
  Regs[R2] = Regs[R2] + d
  Regs[R1] ← Regs[R1] + Mem[Regs[R2]]

- **Autodecrement**: operand is in a memory location specified by a register; register value is predecremented by operand size
  Add R1, -(R2)
  Regs[R2] = Regs[R2] - d
  Regs[R1] ← Regs[R1] + Mem[Regs[R2]]

- **Scaled**: operand is in a memory location specified by two registers, a constant and a scale
  Add R1, 100(R2)[R3]

---

**Addressing Mode Usage In 3 Programs On The VAX Architecture**

- **Displacement Mode**
  - Add R4, 100(R1)
  - How big a displacement should we support?

- **Immediate Mode**
  - Add R4, #3
  - How big a constant should we support?
  - Do we have to support constants for all instructions, or just for some?
Frequency Of Constants On A Benchmark

Constant Size Distribution On A Benchmark

Summary
- GPR architecture, register-register
- Support displacement, immediate and register indirect
- Size of displacement at least 12-16 bits
- Size of immediate at least 8-16 bits

Type And Size Of Operands
- Encoded in opcode
- Type of operand defines its size:
  - Character – 1 byte
  - Half word (short integer) – 2 bytes
  - Word (integer) – 4 bytes
  - Single precision FP and long integer – 4 bytes
  - Double precision FP – 8 bytes
  - Some architectures support packed decimal – two digits per byte

Control Flow Instructions
- Conditional branch
- Unconditional branch: jump
- Procedure call
- Procedure return
Frequency Of Control Flow Instructions On A Benchmark

Addressing Modes For Control Flow Instructions
- Must specify the target address
  - Absolute
  - PC-relative – displacement is added to PC
  - Register-indirect
- PC-relative saves space as target is usually close to branch instruction and provides position independence

Displacement Size Distribution On A Benchmark

How To Specify Branch Condition?
- Condition code
  - ALU operation sets special bits, get condition for free
  - Constrain instruction ordering
- Condition register
  - Write 0 (false) or 1 (true) into a register after comparison
  - Support only BZ and BNZ instructions
- Compare and branch
  - Compare operands (BLT, BGT, BEQ …) and branch
  - Instruction may last long

Procedure Invocation Options
- Return address and some state must be saved
- Caller saving:
  - Calling procedure saves registers that it will need upon return
  - Must be used for globally accessed variables
- Callee saving:
  - Called procedure saves registers that it will overwrite

Summary
- GPR architecture, register-register
- Displacement, immediate and register indirect
- Displacement 12-16 bits, immediate 8-16 bits
- 8-, 16- and 32-bit integers and 32- and 64-bit floating point numbers
- Support displacements of about 100 instructions above or below a branch – 8 bit
- Support PC-relative and register-indirect addressing
Operations in the Instruction Set

- Arithmetic: Add, multiply, subtract, divide
- Logical: And, or
- Control: branch, jump, procedure call and return
- System: OS call, virtual memory management
- FP operations: add, multiply, subtract, divide
- Decimal: add, multiply, convert
- String: move, compare, search
- Graphics: pixel and vertex operations

Homework

- Due Tuesday, 9/21 by the end of the class
- Do exercises 2.3 and 2.6