Instruction Set Operations

- *Arithmetic/Logical : Integer ALU ops.*
    - **ADD** , **AND** , **SUB** , **OR** .

- *Load/Stores : Data transfer between memory and registers.*
    - **LOAD** , **STORE** *(Reg-reg),* **MOVE** *(Mem-mem)*

- *Control : Instructions to change the program execution sequence.*
    - **BEQZ** , **BNEQ** , **JMP** , **CALL** , **RETURN** , **TRAP**

- *System : OS instructions, virtual memory management instructions.*
    - **INT**

- *Floating Point :*
    - **FADD** , **FMULT**

- *Decimal : Support for* **BSD**
- *String : Special instruction optimized for handling ASCII character strings.*
- *Graphics : Pixel operations, compression and decompression.*

Instruction Set Operations

- *All machines generally provide a full set of operations for the first three categories.*

- *All machines MUST provide instruction support for basic system functions.*

- *Floating point instructions are optional but are commonly provided.*

- *Decimal and string instructions are optional but are disappearing in recent ISAs. They can be easily emulated by sequences of simpler instructions.*

- *Graphic instructions are optional.*

- *Remember* **MAKE THE COMMON CASE FAST** *?*
    - *ALU and Load/Store instructions represent a significant portion of the instruction mix and therefore should execute quickly.*

Instruction Set Operations

- *Control Flow instructions:*

    - *Four types are identifiable:*

- *Conditional branches*
- *Jumps*

- *Procedure Calls*
- *Procedure Returns*

- *Program analysis shows that Conditional branches dominate (> 80%).*

- *The destination address must always be specified.*
    - *In most cases, it is given explicitly in the instruction.*
    - *Exception: Procedure return addresses are* **not** *known at compile time.*

Control Flow Operations

- *Addressing Modes:*
- *PC-relative : Most common.*
    - *Constant in instruction gives the offset to be added to the PC.*
    - *Adv:*
        - *Since target is often near the current instruction, the displacement can be small, requiring few address bits.*
        - *Allows relocatable (position independent) code.*

- *Indirect (jump to the address given by a register).*
    - *For procedure returns and indirect jumps for which the address is not known at compile time.*
    -
    - **Register indirect jumps** *useful for three important features:*
- *Case or switch statements.*
- *Dynamic shared libraries*
- *Virtual functions.*

- *Absolute (jump to location in memory). Not commonly used.*

Control Flow Operations

- *Conditional branches:*
    - *Issue: What is the appropriate* **field size** *for the offset ?*
        - *Important because it affects instruction length and encoding.*
    -
- *Observations:*

- *Most frequent branches for integer programs are targets 4 to 7 instructions away (for DLX). This suggests a small offset field is sufficient.*
    -
- *Most* **non-loop** *branches (up to 75% of all branches) are forward. However, they are hard to "predict" and optimize.*
    -
- *Most* **loop** *branches are backward. Backward branches are usually taken, since they are usually part of loops.*

Control Flow Operations

- *Conditional branches: Methods of testing the condition:*

- *Condition Codes (CC) :*
    - *Special bits are set by ALU operations as a side effect.*
    - *Adv:*
        - *Reduces instruction count - it's done for free.*
    - *Disadv:*
        - *Extra state that must be implemented.*
        - *More importantly, it constrains the ordering of instructions (no intervening instructions allowed (that set the CC) between the instruction that sets the CC and the branch that tests the CC).*

- *Condition register :*
    - *Comparisons leave result in a register, which is tested later.*

- *Compare and branch :*
    - *Similar to b, but allows more complex comparisons in the branch.*
    - *Sometimes, this is too complex.*

Control Flow Operations

- **Subroutines** *:*
    - *Include control transfer and return + some state saving.*

- *Should architecture save registers or should compiler do it ?*

- *Caller saving : Caller saves any registers that it wants to use after the call.*
- *Callee saving : Callee saves the registers.*

- *Compiler must determine if procedures may access (global) register-allocated quantities.*
    - *This determination is complicated by separate compilation.*

- **Optimal** *solution in many cases is to have compiler use both conventions.*

- *Implementing sophisticated instructions to do the saving is often in contradiction with the optimal solution.*

Type and Size of Operands and Encoding an Instruction Set

- *Read these sections: we talked about most of this already.*

- **Size and Type of Operands summary** *:*
    - *Frequency of use data indicates that a new 64 bit architecture should support 8 , 16 , 32 and 64 bit integers and 64-bit IEEE 754 floating point data.*

- **Encoding summary** *:*
    - *An architect more interested in code size will pick variable encoding.*

- *Allows virtually all addressing modes in all operations.*
- *This style is best when there are lots of addressing modes and operations.*
- *Instruction differ significantly in the amount of work performed.*

    - *An architect more interested in simplifying instruction decoding in the CPU will pick fixed encoding.*

- *Operation and addressing mode encoded into the opcode.*