Instruction Set Principles

Krerk Piromsopa, Ph.D.
Dept. of Computer Engineering
Chulalongkorn University

Classifying Instruction Set Architectures

- R-R
  - Simple, fixed-length encoding, similar CPI
  - High IC, Lower density, Larger Program
- R-M
  - No separate load, easy encoding, good density
  - Operands not equivalent, Less Regs, vary CPI
- M-M
  - Compact, no waste registers
  - Vary length, vary CPI (high), memory bottleneck

Flynn's Taxonomy

	Single Instruction	**Multiple Instruction**
Single Data	SISD (uniprocessor)	MISD (fault-tolerant)
Multiple Data	SIMD (vector)	MIMD (multiprocessor)

Example

- Several researchers hav suggested that adding a register-memory addressing mode to a load-store machine might be useful. The idea is to replace sequences of

```
LOAD    R1, 0(RB)
ADD R2, R2, R1
```

by

```
ADD R2, 0(RB)
```

Assume the new instruction will cause th e clock cycle to increase by 5% with out affecting the CPU. If 26% of a program is accounted for LOAD instructions and 19% is accounted for ADD instructions,

a. What percentage of the loads must be eliminated for the machine with the new instruction to have at least the same performance?

Example

- Consider this three statements:

```
A=B+C;
B=A+C;
D=A-B;
```

Use the technique of copy propagation to transform the code sequence to the point where no operand is a computed value. Note the instances in which the transformation has reduced the computational work of a statement and those cases where the work has increased. What does this suggest about the technical challenge faced in trying to satisfy the desire for optimizing compilers?

Example

- Optimize the following code:

```
for(x=0;x<=100;x++) {
   hs=3600;
   ds=hs*24;
   h=a[x]/hs;
   d=a[x]/ds
}
```