

# **III. Interpreters and Compilers**

# Translators

- A translator is a program that accepts any text expressed in one language (the translator's **source language**) and generates a semantically equivalent text expressed in another language (its **target language**)
- Examples:
  - A natural language translator (Navajo into English) [not the topic of this course]
  - A C++ into SPARC machine language translator.
- Classes of programming language translators include

- **Assemblers:** From assembly language to machine code
- **Compilers:** High-level languages to assembly/ machine language (Conventional C compiler) or to low-level code (some Pascal compilers translated into **P-code**, many Java compilers translate into **byte code**).
- **Source-to-source translators:** Both source and target are high-level languages. In many cases, both target and source are ascii streams (For example, KAI's C++ to C translator), but also many translators use internal representations as its target.
  - > Examples of source-to-source translators include **Preprocessors** (macro expanders) and **Parallelizers** (which transform sequential programs into parallel form).

- > Internal representation is a data structure (for example a tree) that represents the organization of the source program in a way that is easy to manipulate by a translator or interpreter.
- Translators have several goals:
  - Facilitate programming by enabling code development using a high-level language.
  - Checking correctness of the source program
    - > For example, indicating a syntax error if the program contains the addition of a number and something that is not (and cannot be transformed into) a number)
  - Generating fast code by applying optimizations (see next two examples).

q

## Consider the loop

```
do i=1,n
  s=s+b(i)
end
```

### A naive translation

```

      l   R1,#1
      l   R1,#1
L1:   cmp R1,n
      bgt L2
      l   R2,R1
      sub R2,#1
      mpy R2,#4
      add R2,#b
      l   F1,(R2)
      l   F2,s
      add F1,F2
      st  F1,s
      add R1,#1
      br  L1
L2:
```

### A better translation

```

      l   F1,s
      l   R2,n
      sll R2,n
      add R2,#b
      l   R1,#b
      cmp R1,R2
      bgt L2
L1:   add F1(R2)
      add R2,#4
      cmp R2,R1
      blt L1
L2:   st  F1,s
```

The following example of code improvement in source-to-source translation deals with locality enhancement.

Consider the following matrix multiplication loop

```
do i=1 to n by 1
  do j=1 to n by 1
    c(i,j)=0;
    do k=1 to m by 1
      c(i,j)=c(i,j)+a(i,j)*b(k,j)
    end
  end
end
```

Would perform much better in a paging environment if written as shown next.

```

instart=1
inend=m
indiff=1
do baserow=1 to 1 by rpp
  lastrow=max(baserow+rpp-1,1)
  do i=baserow to lastrow by 1
    do j=1 to n by 1
      c(i,j)=0
    end
  end
  do k=instart to inend by indiff
    do i=baserow to lastrow by 1
      do j=1 to n by 1
        c(i,j)=c(i,j)+a(i,k)*b(k,j)
      end
    end
  end
end
temp=instart
instart=inend
inend=temp
indiff=-indiff
end

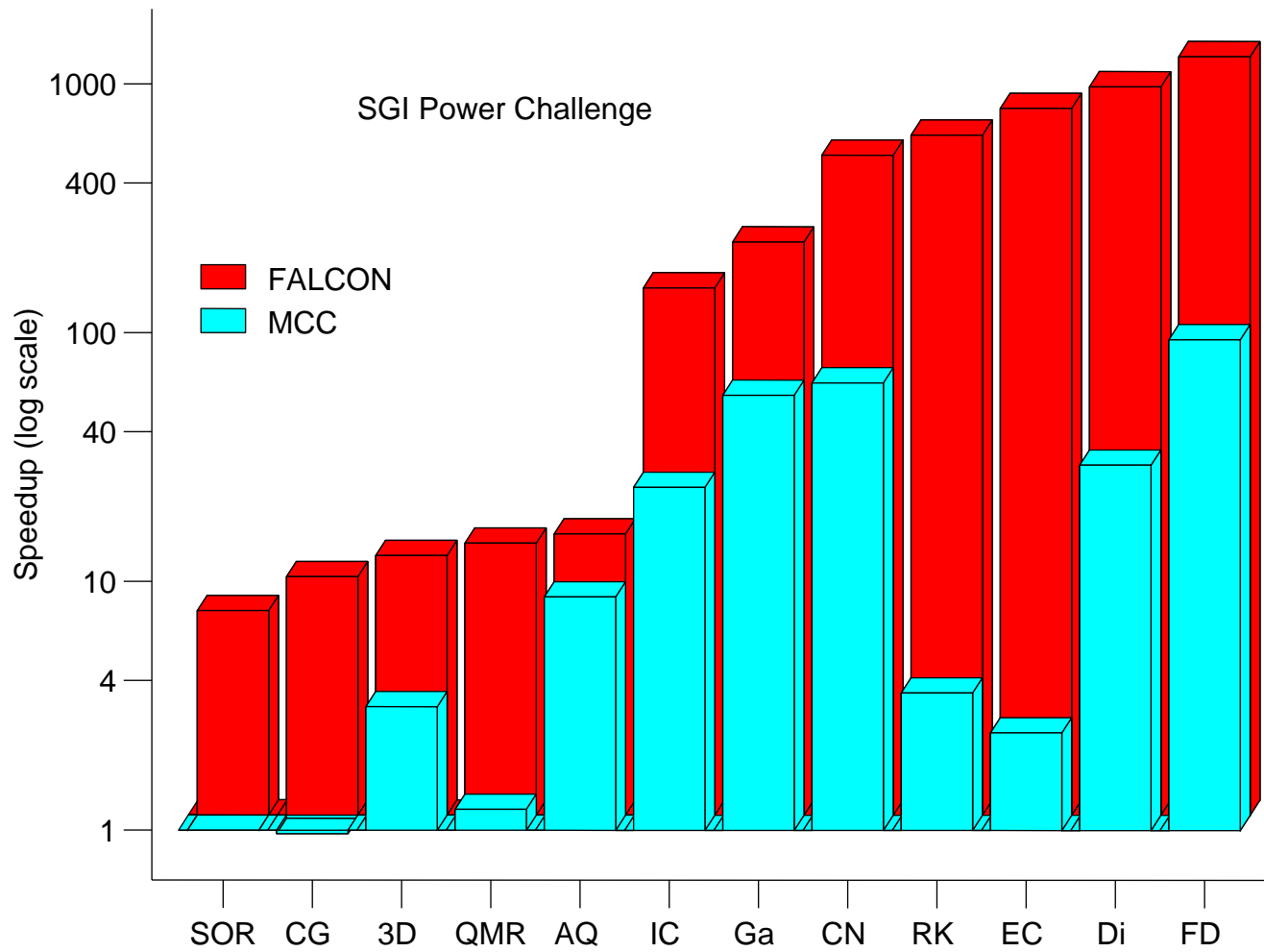
```

# Interpreters

- Are the devices that actually execute the program.
  - Simplest interpreter is a regular computer system. It takes machine language and executes it.
  - A JVM (Java Virtual Machine) interpreter is a program that takes byte code and executes it.
  - The MATLAB interpreter takes (the internal form) of the source code and executes it.
- High-level language interpreters facilitate debugging and program development in general, but performance can be dismal.

- For example, translating MATLAB into Fortran 90 and then translating Fortran 90 into machine language for the SGI produces programs that execute much faster than purely interpreted MATLAB codes.

-



- Interpretation usually involves several layers.
  - A JVM interpreter is itself interpreted by the machine in which it is running.
  - Even the interpretation of machine code can involve several layers if the machine is microprogrammed. The microprogram interprets the machine language program and the CPU interprets the microprogram.
- In most cases, both interpretation and compilation are needed for the execution of programs.
  - MATLAB translates into (high-level) internal form and then interprets the program.
  - Fortran and C formats are interpreted.
  - Most JVM interpreters compile on-the-fly (or Just-In-Time)