HPC Fundamentals
Programming Languages

Robert Whitten Jr

Programming Languages

- Computers execute instructions, i.e.:
  - Move data to a memory location
  - Add data in one memory location to another
  - Jump to next instruction to execute
- Programming languages allow the order of instructions to be controlled
  - Each type of processor has different instructions
- Low-level vs. high-level languages
  - Machine languages
  - Assembly languages
  - High-level languages
- Interpreted vs. compiled

Machine languages

- Its what computers understand
  - Low level language
  - Specific to a particular architecture
    - x86, IA-32, x86-64, AMD64, Motorola's 6800 and 68000
- Binary instructions
  - General form

6 5 5 5 5 6 bits

[op | rs | rt | rd |shamt| funct] R-type

[op | rs | rt | address/immediate] I-type

[op | target address] J-type

- Example

000010 00000 00000 00000 10000 000000

Assembly language

- One-to-one mapping to machine instruction
- Specific to each architecture
  - Usually provided by manufacturer or processor
- Low-level language
- Uses mnemonics representation of instructions
  - mv ax, es
  - add ip, ax
- Example:

Machine code= 000010 00000 00000 00000 10000 000000

Assembly = jmp 1024

High-level languages

- Abstract representation of program
  - Details of architecture are hidden
  - More portability
  - Easier to program
- Structured for humans not machines
- Examples
  - C++
  - Fortran
  - Java
  - Python
  - Perl
  - Hundreds exist

Interpreted vs. compiled vs. markup

- Interpreted languages
  - Require special programs that execute other programs
  - Typically 1 line at a time
  - Shell script, Python, Perl
- Compiled languages
  - Do not require a separate program to execute
  - Does require a program called a compiler
  - Typically faster than interpreted languages (not always)
  - C/C++, Fortran
- Markup languages
  - Not really a programming language (a way to format data)
  - HTML, XML, SGML

Interpreted languages

- Start with a text file (call source file)
  - 1st line is typically #!<path to interpreter program>
    - #!/bin/bash or #!/bin/perl, etc.
- Interpreter reads and executes one line at a time
  - Syntax errors are caught immediately
  - Does not look forward to see what's next
- Programmers can usually test singe statements in interpreter
  - echo "hello"

Compiled languages

- Starts with text file (source code)
- A compiler program:
  - Coverts source into assembly language
  - Assembly is converted into machine code (object code)
  - Object code is linked with other object code to make executable by linker program (often part of compiler
- Final code does not need an interpreter to execute
  - Can execute on own
    - ./programName
- Typically faster than interpreted languages
  - Doesn't require overhead of running another program

Markup languages

- Not a programming language
  - The end result is not executable code
- Designed to format data to make presentable
  - Bold, italic, justified, formated in a table, etc
- HTML
  - Hypertext Markup Language
- XML
  - eXtensible Markup Language
- SGML
  - Standard Generalized Markup Language
  - Grandparent to HTML and XML

C basics

- Comments
- Variables
- Constants
- Selection
- Loops
- Functions

C program example – Hello World

```
#include <stdio.h>


void main(void)
{
    printf(“Hello World\n”);
}
```

Hello World +

```
#include <stdio.h>
void main(void)
{
    int x;
    x=10;
    printf("Hello World\n");
    printf("x=%d",x);
}
```

Hello World ++

```
#include <stdio.h>


int main(int argc, char** argv)
{
    printf("Hello World, %s\n", argv
 [1]);
    return (0);
}
```

Variables

Type	Size	Range
unsigned char	8 bits	0 to 255
char	8 bits	-128 to 127
unsigned int	16 bits	0 to 65,535
short int	16 bits	-32,768 to 32,767
int	16 bits	-32,768 to 32,767
unsigned long	32 bits	0 to 4,294,967,295
long	32 bits	-2,147,483,648 to 2,147,483,647
float	32 bits	1.17549435 * (10^-38) to 3.40282347 * (10^+38)
double	64 bits	2.2250738585072014 * (10^-308) to 1.7976931348623157 * (10^+308
long double	80 bits	3.4 * (10^-4932) to 1.1 * (10^4932)

Variable declaration

- int I;
- char c;
- double dbl;
- float f;
- int i=0;
- const pi=3.14159265;

Arrays

- One dimensional
  - int i [10];
  - char str [25];
- Two dimensional
  - int d [2][2];

Selection

- if condition then something else something else.

```
if (x < 10)
{
  printf("low\n");
}
```

Loops

- While condition do

```
while (x < 10)
{
  printf("low\n");
  x++;
}
```

- Do until condition

```
do
{
  printf("low\n");
  x++;
} while (x<10);
```

Loops

- For expression do

```
for (i=0; i<10; i++)
{
  printf("low\n");
}
```

Functions

- Recurring pieces of code

```
<return> func (parameters)
{
 body
}
```

Exercise

- Write a C program
- The program should perform a matrix multiplication of an MxM matrix
  - Many scientific application rely heavily on linear algebra
    - Matrix multiplication is one such algorithm

Matrices: What They Are

$$\begin{matrix} 4 & 2 & 2 & 1 \\ 1 & 1 & 1 & 1 \\ 2 & 2 & 1 & 1 \\ 2 & 3 & 5 & 6 \end{matrix}$$

- Multidimensional grids of numbers
- Properties
  - Can be rectangular
  - Can be 1D, 2D,3D, 4D, …
  - m x m, m x n

Matrices and Modeling

4	2	2	1
1	1	1	1
2	2	1	1
2	3	5	6

- Matrix models related set of measures: e.g.,
  - Counts
  - Costs
  - Physical data
- Measures characterize a set of related, measurable entities (e.g., people)
- Columns model properties of these entities

Matrices and Modeling (example)

4	2	2	1
1	1	1	1
2	2	1	1
2	3	5	6

- Matrix counts social activities
- Rows represent set of friends
  - Lynn, Mel, Noel, Pat
- Columns represent activity types
  - Col. 1: movies
  - Col. 2: dinners
  - Col. 3: car trips
  - Col. 4: gifts

Matrices and Modeling (example)

$$\begin{matrix} 4 & 2 & 2 & 1 \\ 1 & 1 & 1 & 1 \\ 2 & 2 & 1 & 1 \\ 2 & 3 & 5 & 6 \end{matrix}$$

- Matrix models cost of social activities (10's of $s)
- Rows represent set of friends
  - Lynn, Mel, Noel, Pat
- Columns represent activity types
  - Col. 1: movies
  - Col. 2: dinners
  - Col. 3: car trips
  - Col. 4: gifts

Matrices and Modeling (example)

4	2	2	1
1	1	1	1
2	2	1	1
2	3	5	6

- Matrix models value of financial instruments (1,000s of $s)
- Rows represent institution's depositors
- Columns represent instrument type (1,000's of $s)
  - Col. 1: stocks
  - Col. 2: bonds
  - Col. 3: commodities
  - Col. 4: cash

Matrices and Modeling (example)

4	2	2	1
1	1	1	1
2	2	1	1
2	3	5	6

- Matrix models counts of registered voters
- Rows represent locales of interest
- Column represent classes of registrations
  - Col. 1: Republicans
  - Col. 2: Democrats
  - Col. 3: Independents
  - Col. 4: Unregistered

Matrices and Modeling (example)

4	2	2	1
1	1	1	1
2	2	1	1
2	3	5	6

- Matrix models counts of birds
- Rows represent species
- Columns represent counting stations

Matrices and Modeling (example)

4	2	2	1
1	1	1	1
2	2	1	1
2	3	5	6

- Matrix models ocean temperatures (degrees Celsius)
- Rows represent buoys
- Columns represent points in time
  - Col. 1: 18 hours ago
  - Col. 2: 12 hours ago
  - Col. 3: 6 hours ago
  - Col. 4: now

Matrices and Modeling

4	2	2	1
1	1	1	1
2	2	1	1
2	3	5	6

- Can flip-flop row, column semantics for these examples, e.g.,
  - Columns represent buoys
  - Rows represent points in time

Matrices and Modeling (example)

	movies	dinners	car trips	gifts
Lynn	4 movies	2 din.	2 car t.	1gift
Mel	1 movie	1 din.	1car t.	1gift
Noel	2 movies	2 din.	1car t.	1gift
Pat	2 movies	3 din.	5 car t.	6 gifts

- If movies cost $10, dinners cost $20, car trips cost $30, and gifts cost $40, how much did Lynn, Mel, Noel, and Pat spend for the indicated time frame?

Solution (Vector-Matrix Product)

$$
\begin{vmatrix}
4\text{ mv.} & 2\text{ din.} & 2\text{ cr t.} & 1\text{ gifts} \\
1\text{ mv} & 1\text{ din.} & 1\text{ cr t.} & 1\text{ gift} \\
2\text{ mv} & 2\text{ din.} & 1\text{ cr t.} & 1\text{ gift} \\
2\text{ mv} & 3\text{ din.} & 5\text{ cr t.} & 6\text{ gifts}
\end{vmatrix}
*
\begin{vmatrix}
\$10/\text{mv.} \\
\$20/\text{din.} \\
\$30/\text{cr t.} \\
\$40/\text{gift}
\end{vmatrix}
=
$$

$$
\begin{vmatrix}
4\text{ mv.}*\$10/\text{mv.} + 2\text{ m.o.}*\$20/\text{m.o.} + 2\text{ cr t.}*\$30/\text{cr t.} + 1\text{ gift}*\$40/\text{gift} \\
1\text{ mv.}*\$10/\text{mv.} + 1\text{ m.o.}*\$20/\text{m.o.} + 1\text{ cr t.}*\$30/\text{cr t.} + 1\text{ gift}*\$40/\text{gift} \\
2\text{ mv.}*\$10/\text{mv.} + 2\text{ m.o.}*\$20/\text{m.o.} + 1\text{ cr t.}*\$30/\text{cr t.} + 1\text{ gift}*\$40/\text{gift} \\
2\text{ mv.}*\$10/\text{mv.} + 3\text{ m.o.}*\$20/\text{m.o.} + 5\text{ cr t.}*\$30/\text{cr t.} + 6\text{ gifts}*\$40/\text{gift}
\end{vmatrix}
=
$$

$$
\begin{vmatrix}
\$40 + \$40 + \$\ 60 + \$\ 40 \\
\$10 + \$20 + \$\ 30 + \$\ 40 \\
\$20 + \$40 + \$\ 30 + \$\ 40 \\
\$20 + \$60 + \$150 + \$240
\end{vmatrix}
\begin{vmatrix}
\$180 \\
\$100 \\
\$130 \\
\$470
\end{vmatrix}
\begin{matrix}
\text{(Lynn)} \\
\text{(Mel)} \\
\text{(Noel)} \\
\text{(Pat)}
\end{matrix}
$$

Notation

- Matrix quantities typically in UPPER CASE
- Vector quantities typically in lower case
- Denoting

$$\begin{vmatrix} 4 & 2 & 2 & 1 \\ 1 & 1 & 1 & 1 \\ 2 & 2 & 1 & 1 \\ 2 & 3 & 5 & 6 \end{vmatrix}$$ by EVENTS, $$\begin{vmatrix} 10 \\ 20 \\ 30 \\ 40 \end{vmatrix}$$ by costs,

we have EVENTS X costs= $$\begin{vmatrix} 180 \\ 100 \\ 130 \\ 470 \end{vmatrix}$$

Matrix multiply

- A * B = C

$$\begin{vmatrix} a_{i0j0} & a_{i0j1} \\ a_{i1j0} & a_{i1j1} \end{vmatrix} \begin{vmatrix} b_{i0j0} & b_{i0j1} \\ b_{i1j0} & b_{i1j1} \end{vmatrix} = \begin{vmatrix} c_{i0j0} & c_{i0j1} \\ c_{i1j0} & c_{i1j1} \end{vmatrix}$$

$$\begin{vmatrix} a_{i0j0} & a_{i0j1} \\ a_{i1j0} & a_{i1j1} \end{vmatrix} \begin{vmatrix} b_{i0j0} & b_{i0j1} \\ b_{i1j0} & b_{i1j1} \end{vmatrix} = \begin{vmatrix} c_{i0j0} & c_{i0j1} \\ c_{i1j0} & c_{i1j1} \end{vmatrix}$$

$$\begin{vmatrix} 3 & 2 \\ 4 & 3 \end{vmatrix} \begin{vmatrix} 1 & 3 \\ 2 & 1 \end{vmatrix} = \begin{vmatrix} (3*1)+(2*2) & (3*3)+(2*1) \\ (4*1)+(3*2) & (4*3)+(3*1) \end{vmatrix}$$

$$\begin{vmatrix} 7 & 11 \\ 10 & 15 \end{vmatrix}$$

Questions?

http://www.olcf.ornl.gov