Language of Computer?

- Executes instructions (e.g., add, multiply) on registers

```
add      dst    src1    010    src2
```

- Move memory contents to registers

```
ldr      dst    base    offset
```

- Change Program Counter (PC) to new location

```
jmp      base    00000000
```
Why Mismatch between Computer and Programmer?

Disadvantages

- Every architecture has own machine language
  - Program not “portable”; must rewrite for every platform
- Programmer must write lots of statements
  - Need lots of add, multiply, etc. to do anything useful
- Programmer must track location of every variable and instruction in memory
- Time consuming, tedious, error prone

What do Programmers Want?

Portability: Runs across multiple architectures

- Tuned and optimized for different platforms!

Expressibility: Describe algorithm easy for human

- Similar to “natural” language or standard mathematical notation

Productivity: Write few statements quickly

Eliminate errors: Avoid programming mistakes

- Don’t require programmer to manage details of memory movement

Performance: Runs quickly and space efficiently
Many Different Programming Languages

C Java C++ PHP (Visual) Basic C# Python Perl
Delphi JavaScript Objective-C Ruby SAS
PL/SQL Go Pascal ABAP MATLAB ActionScript
Lua Transact-SQL JavaFX Lisp/Scheme Scratch D
Bourne shell Scala Erlang COBOL RPG S-lang Logo Ada
Fortran NXT-G Tcl/Tk Alice Forth FoxPro/Caml/F#
PowerShell Prolog Groovy Smalltalk VHDL J C shell MAX/MSP VBScript AD, Algol, Alpha, APL, Applescript, Awk, Beta, cg, Clean, cT, Dylan, Eiffel, Factor, Falcon, Focus, Haskell, Heron, Icon, IDL, Informix-4GL, Io, LabVIEW, LabWindows/CVI, Lingo, MAD, Maple, Mathematica, ML, Modula-2, Modula-3, MOO, MS-DOS batch, MUMPS, Natural, Occam, Oz, PL/I, Postscript, PowerBuilder, Progress, Q, R, REALbasic, Revolution, REXX, SIGNAL, SPSS, SuperCollider, Verilog, XSLT

Why So Many Languages?

Different languages have different strengths
Match to specific domains
• Web applications, databases, business, numerical, parallelism
Trade-off in development time, performance, error cases
Development time
• How quickly can you go from new idea to running code?
  − How much to type? Don’t need to declare type of variables, length of lists
  − Wait to compile? Interpret code instead
Performance
• More effort up front can lead to better performance in long run
  • E.g. Manage memory by hand (C) vs garbage collection (Java)
Avoiding errors
• Catch errors when compile or at run-time?
• Maintaining code base of large multi-person projects
Classes of Programming Languages

Procedural/Imperative (e.g., C, FORTRAN, Pascal, BASIC)
- Most closely matches machine language
  - Variables represent locs in memory, statements op on variables
  - Can map program instructions to machine instructions
- Variant: Object-oriented (e.g., Java, C++, Smalltalk, Scratch?)
  - Objects have associated variables and code

Logical/Declarative (e.g., Prolog, SQL)
- Set of rules or logic; execution finds proof satisfying rules
  - State what programmer wants done, not how to do it!

Functional (e.g., Lisp, Scheme, Haskell)
- Allows functions to be used as data in computation
  - Popular in AI

How does Programming in Scratch Compare to C?

```c
#include <stdio.h>

int main(int argc, char *argv[])
{
    printf("Hello World!\n");
}
```

Compile text file “hello.c” into executable file named “hello” for this architecture
Can execute “hello” directly on this architecture
Can specify optimization level
  - Spend longer time compiling
  - Faster execution later
Variables and If-Else Control Flow

```c
int main(int argc, char *argv[]) {
    int fav_class;
    printf("Favorite CS class?\n");
    scanf("%d", &fav_class);

    if (fav_class == 202) {
        printf("Great Semester!\n");
    } else if (fav_class == 302) {
        printf("Programming is fun!\n");
    } else if (fav_class == 252) {
        printf("Great!  Hardware!\n");
    } else {
        printf("%d?? Tell me more about that class!\n", fav_class);
    }
}
```

C: Must declare TYPE (integer, float, character) of variables
All languages typically have way to format output...

While vs. Repeat Until Loops

```c
int main(int argc, char *argv[]) {
    int i;
    i = 1;
    while (i <= 10) {
        printf("%d: Hello World\n", i);
        i = i+1;
    }
}
```

While loop executes "as long as" test condition is true
For Loops

```c
int main(int argc, char *argv[]) {
    int i;
    i = 1;
    while (i <= 10) {
        printf("%d: Hello World\n", i);
        i = i+1;
    }
}
```

Simplify common structure

For loop: Three parts
- Initialization code, test of when loop should terminate, and code for loop iterator

Procedures vs. Scripts

```c
int main(int argc, char *argv[]) {
    int a, b, result;
    printf("Enter an integer.\n"); scanf("%d", &a);
    printf("Enter another int.\n"); scanf("%d", &b);
    result = mystery(a, b);
    printf("Result is %d\n", result);
}
```

```c
int mystery(int a, int b) {
    int i, answer = 1;
    for (i = 0; i < b; i++) {
        answer *= a;
    }
    return answer;
}
```

Receiving a message is similar to calling procedure

Procedures have better support for passing parameters and returning results
Lists vs. Arrays

float CalculateTotal()
{
    float Total = 0;
    int i;

    for (i = 0; i < LIST_SIZE; i++) {
        Total += Prices[i];
    }

    return Total;
}

Arrays in C start with element number 0

Access element 0 of array with notation Array[0]

Can't dynamically ask for length!

Arrays vs Lists

void FindBargains()
{
    int i;
    int j;

    for (i=0, j=0; i<LIST_SIZE; i++){
        if (Prices[i] <= 5.00) {
            BudgetPrices[j] = Prices[i];
            BudgetFood[j]=GroceryList[i];
            j++;
        }
    }
}
Arrays vs. List

```c
result = LeastExpensive();
printf("The cheapest item is %s for %.2f\n", GroceryList[result], Prices[result]);
```

```c
int LeastExpensive()
{
    int i;
    float min = Prices[0];
    int result = -1;
    for (i=0; i < LIST_SIZE; i++)
    {
        if (Prices[i] < min) {
            min = Prices[i];
            result = i;
        }
    }
    return result;
}```

What is Scratch Missing?

Scratch limits functionality to avoid overwhelming beginners
Procedures, functions (pass/return variables)
File Input/Output
  • Can’t manipulate data sets
Dynamically allocate variables and Sprites
  • Often don’t know before program begins how many needed (depends on user’s actions)
  • Inheritance: Sprites inherit some scripts, redefine others
Complex data structures
  • Lists of lists (2-D arrays); Records
Other Courses?

252 Introduction to Computer Engineering 2 cr
• Logic components built with transistors, Boolean algebra, combinational logic design, synchronous sequential logic design, computer organization and design, machine-and assembly-language programming

302 Introduction to Programming 3 cr
• Instruction and experience in Java, an object-oriented programming language. Program design; development of good programming style

Introduction to Human-Computer Interaction 3 cr
• User-centered design, evaluation methods and their applications in human-computer interaction (HCI) practice. Designing software-products and services for web-based, mobile, and sociable technology.

CS Certificate

CS 202 counts toward Certificate in CS

Requirements: 6 undergraduate courses in CS
• CS302 and CS367
• At most 4 courses below the 400 level
• At least 2 courses at the 400 level or above

Flyers available if interested!
Outreach Opportunities

Teach Scratch programming to 4th/5th graders
• Shorewood Elementary
Every week (Tue or Thu), 3:15 – 4:30
• Lesson to whole group : 20 students
• Individual work time: 1 volunteer / 3-4 students
• Last year: 5 volunteers + Me and Evelyn Eastmond
• Kids did impressive projects! (Didn’t learn Lists!)
Small amount of funding for XO maintenance
• Volunteer, course credit

Today’s Summary

Programming Languages
• Attempt to match programmer and machine
• Variety of requirements lead to different languages
• Learn more about other languages!

Announcements
• Project 3: Due Wednesday – In-class demo
• Friday: Review for Final Exam
• Sunday 10:05 – 12:05: Final Exam