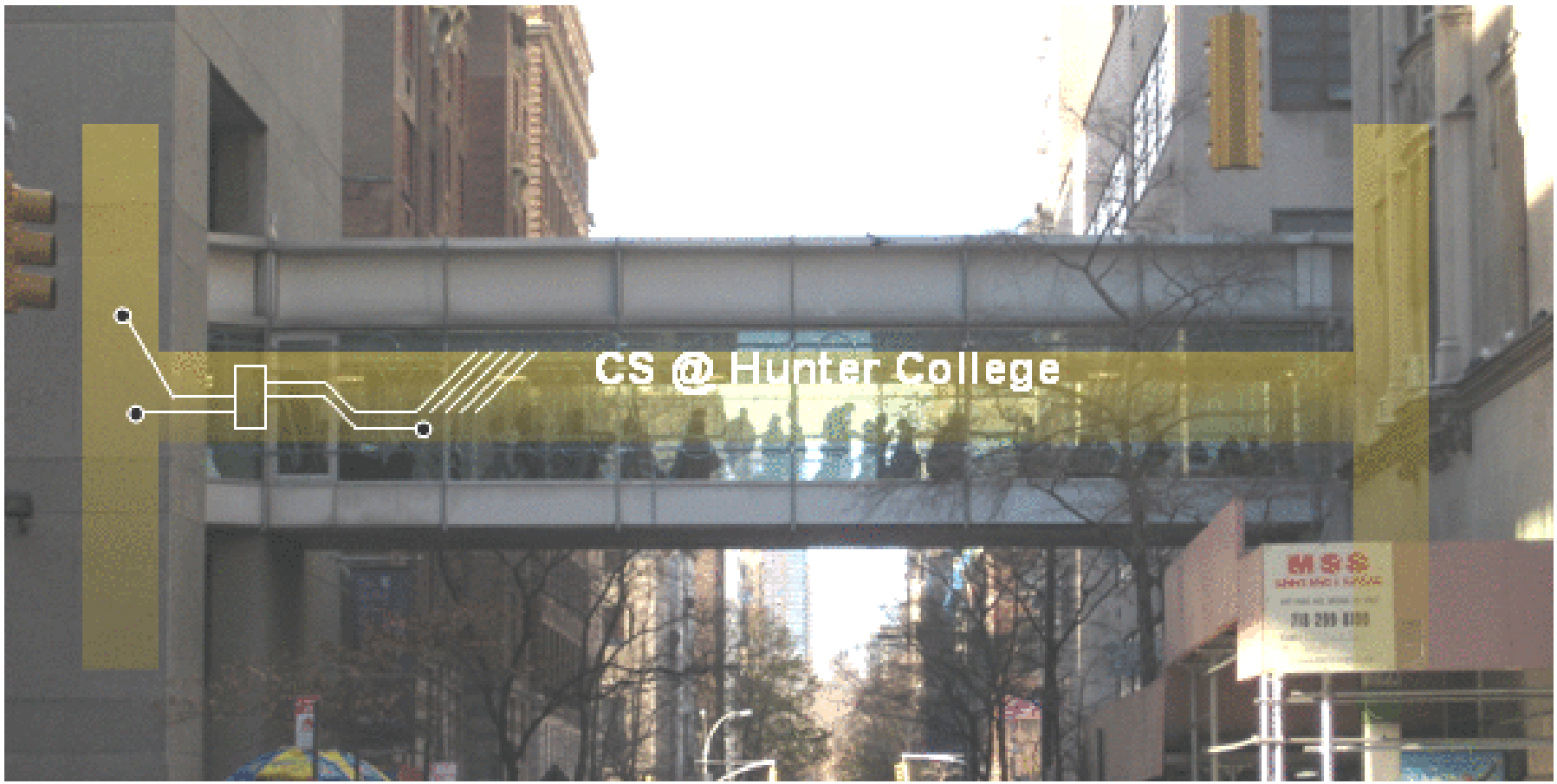
CSci 127: Introduction to Computer Science

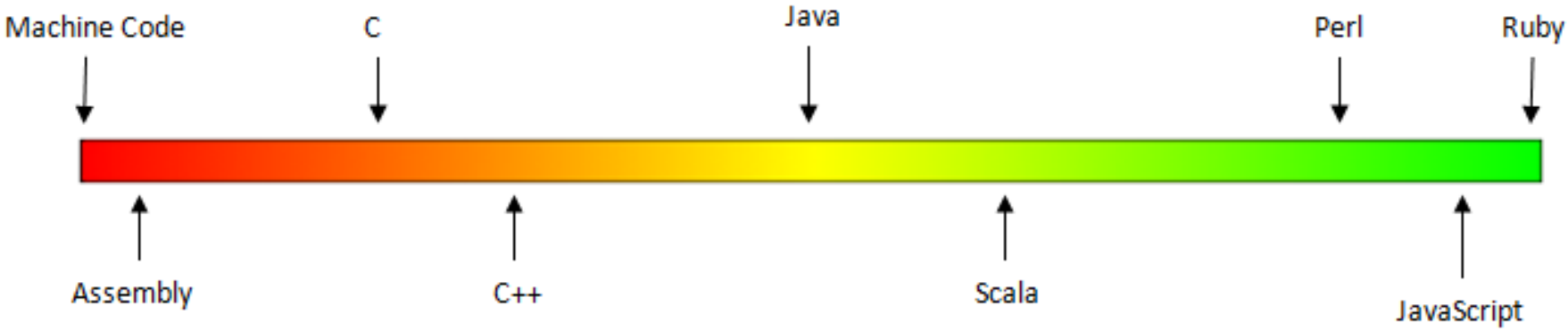


hunter.cuny.edu/csci

Today's Topics

- Recap of Low-Level Programming
- Introducing C++
- Hello, World in C++
- I/O and Definite Loops in C++
- Final Exam Overview

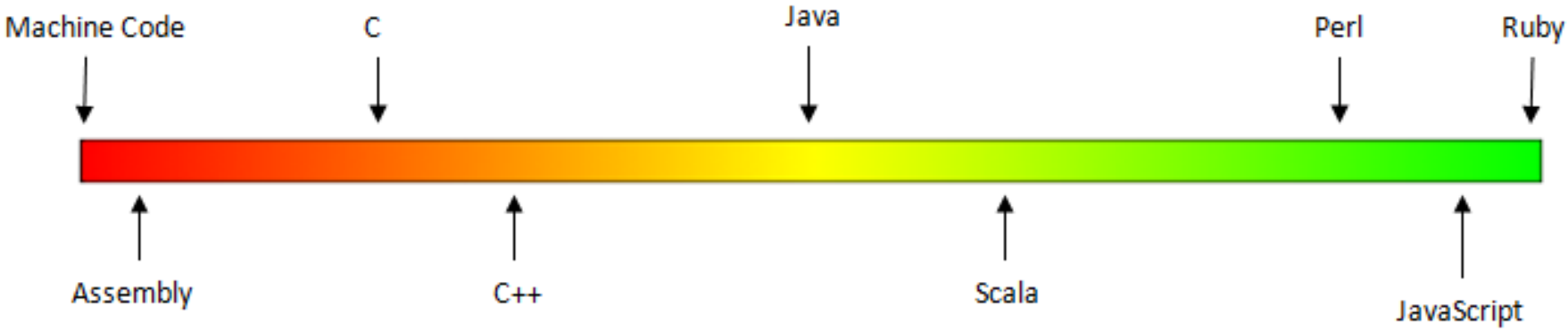
Low-Level vs. High-Level Languages



(codeCommit)

- Can view programming languages on a continuum.

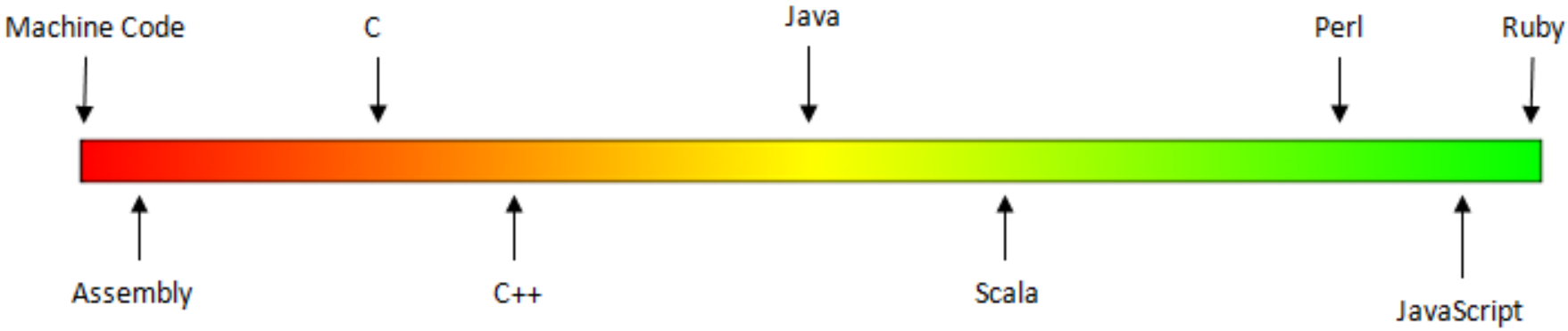
Low-Level vs. High-Level Languages



(codeCommit)

- Can view programming languages on a continuum.
- Those that directly access machine instructions & memory and have little abstraction are **low-level languages**

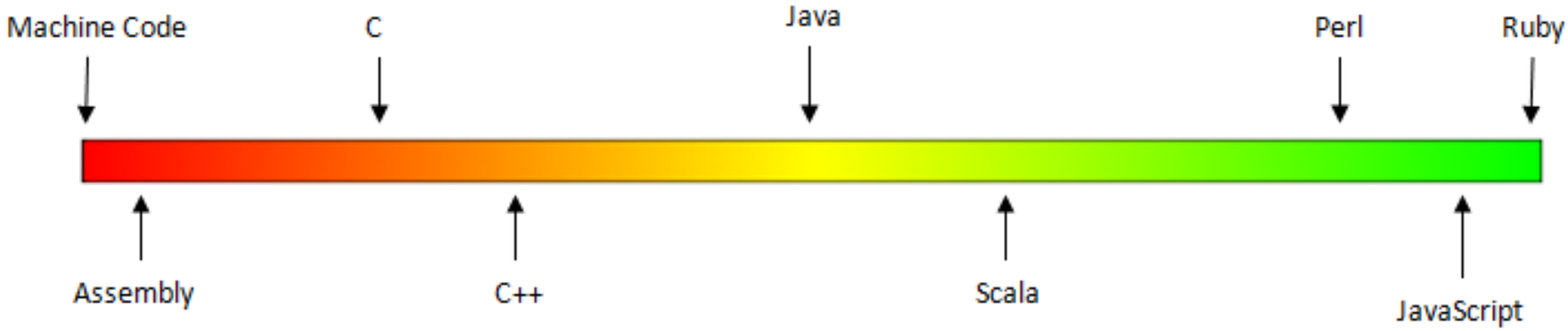
Low-Level vs. High-Level Languages



(codeCommit)

- Can view programming languages on a continuum.
- Those that directly access machine instructions & memory and have little abstraction are **low-level languages** (e.g. machine language, assembly language).

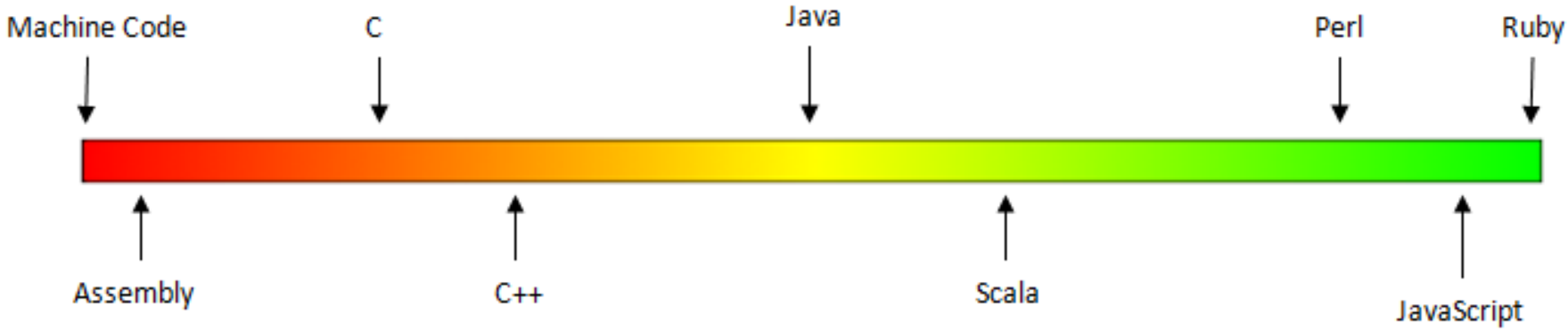
Low-Level vs. High-Level Languages



(codeCommit)

- Can view programming languages on a continuum.
- Those that directly access machine instructions & memory and have little abstraction are **low-level languages** (e.g. machine language, assembly language).
- Those that have strong abstraction (allow programming paradigms independent of the machine details, such as complex variables, functions and looping that do not translate directly into machine code) are called **high-level languages**.

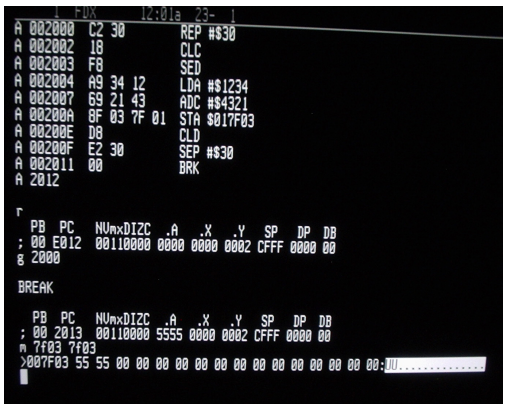
Low-Level vs. High-Level Languages



(codeCommit)

- Can view programming languages on a continuum.
- Those that directly access machine instructions & memory and have little abstraction are **low-level languages** (e.g. machine language, assembly language).
- Those that have strong abstraction (allow programming paradigms independent of the machine details, such as complex variables, functions and looping that do not translate directly into machine code) are called **high-level languages**.
- Some languages, like C, are in between– allowing both low level access and high level data structures.

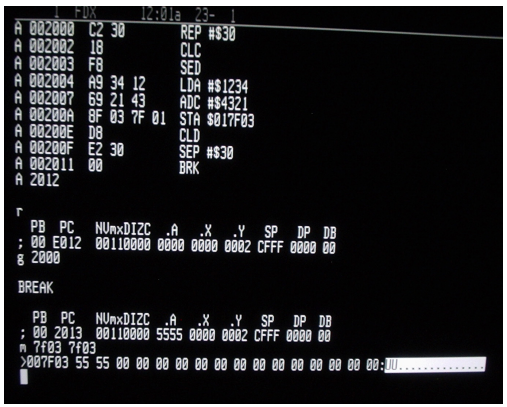
Machine Language



(wiki)

- We will be writing programs in a simplified machine language, WeMIPS.

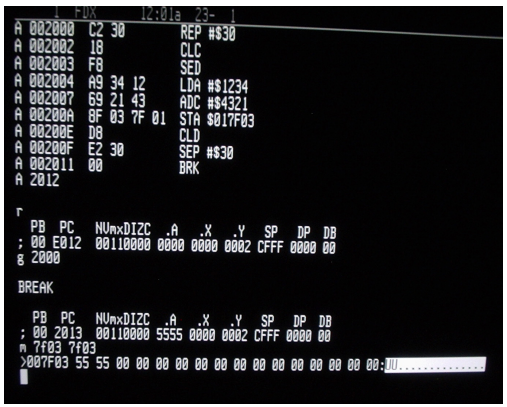
Machine Language



(wiki)

- We will be writing programs in a simplified machine language, WeMIPS.
- It is based on a reduced instruction set computer (RISC) design, originally developed by the MIPS Computer Systems.

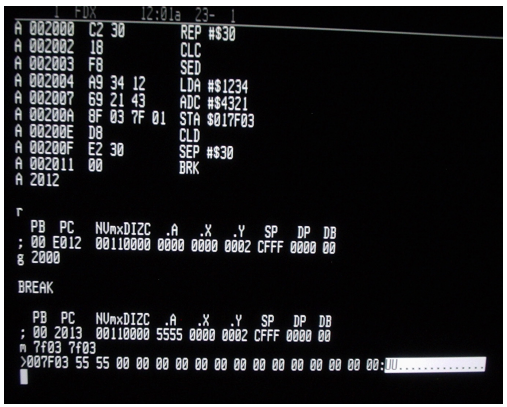
Machine Language



(wiki)

- We will be writing programs in a simplified machine language, WeMIPS.
- It is based on a reduced instruction set computer (RISC) design, originally developed by the MIPS Computer Systems.
- Due to its small set of commands, processors can be designed to run those commands very efficiently.

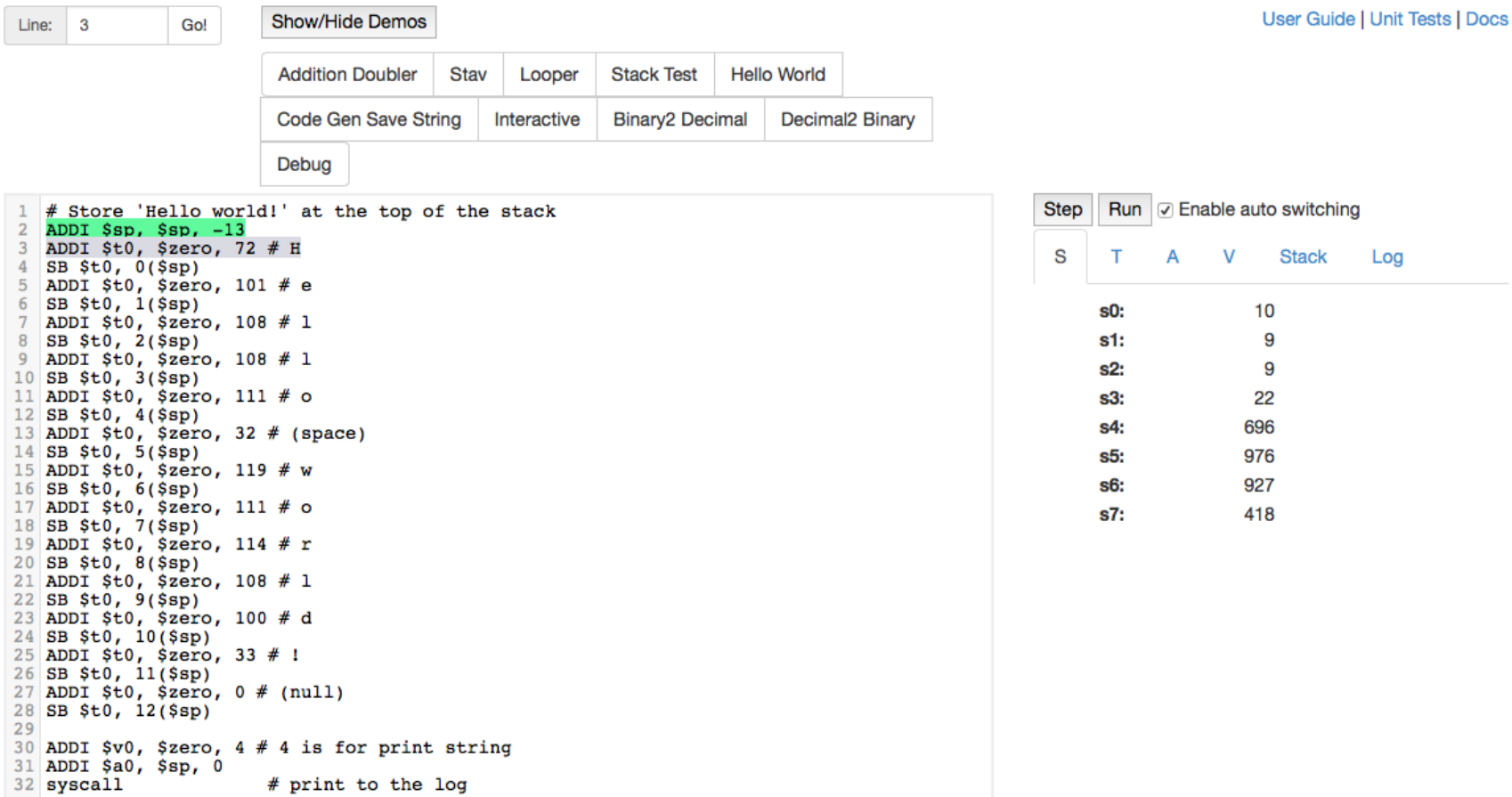
Machine Language



(wiki)

- We will be writing programs in a simplified machine language, WeMIPS.
- It is based on a reduced instruction set computer (RISC) design, originally developed by the MIPS Computer Systems.
- Due to its small set of commands, processors can be designed to run those commands very efficiently.
- More in future architecture classes....

"Hello World!" in Simplified Machine Language



(WeMIPS)

In Pairs or Triples:

Predict what the code will do:

```
1  # This is the same as the doubler, except the jumps cause the order
2  # to change drastically, therefore all of the values will be different.
3  CHANGE_S: ADDI $t0, $zero, 2
4  BEQ $s0, $t0, EXIT
5  ADD $s1, $s0, $s0 # double s0 by adding it to itself, should be 4
6  ADD $s2, $s1, $s1 # double s1 by adding it to itself, should be 8
7  ADD $s3, $s2, $s2 # double s2 by adding it to itself, should be 16
8  ADD $s4, $s3, $s3 # double s3 by adding it to itself, should be 32
9  ADD $s5, $s4, $s4 # double s4 by adding it to itself, should be 64
10 ADD $s6, $s5, $s5 # double s5 by adding it to itself, should be 128
11 ADD $s7, $s6, $s6 # double s6 by adding it to itself, should be 256
12 J CHANGE_V
13
14 CHANGE_T: ADD $t0, $s7, $s7
15 ADD $t1, $t0, $t0
16 ADD $t2, $t1, $t1
17 ADD $t3, $t2, $t2
18 ADD $t4, $t3, $t3
19 ADD $t5, $t4, $t4
20 ADD $t6, $t5, $t5
21 ADD $t7, $t6, $t6
22 ADD $t8, $t7, $t7
23 ADD $t9, $t8, $t8
24 J CHANGE_S
25
26 CHANGE_A: ADD $a0, $t9, $t9
27 ADD $a1, $a0, $a0
28 ADD $a2, $a1, $a1
29 ADD $a3, $a2, $a2
30 J CHANGE_S
31
32 CHANGE_V: ADD $v0, $a3, $a3
33 ADD $v1, $v0, $v0
34 J CHANGE_A
```

WeMIPS

```
# This is the same as the doubler, except the jumps cause the order
# to change drastically, therefore all of the values will be different.
CHANGE_S: ADDI $t0, $zero, 2
BEQ $s0, $t0, EXIT
ADD $s1, $s0, $s0 # double s0 by adding it to itself, should be 4
ADD $s2, $s1, $s1 # double s1 by adding it to itself, should be 8
ADD $s3, $s2, $s2 # double s2 by adding it to itself, should be 16
ADD $s4, $s3, $s3 # double s3 by adding it to itself, should be 32
ADD $s5, $s4, $s4 # double s4 by adding it to itself, should be 64
ADD $s6, $s5, $s5 # double s5 by adding it to itself, should be 128
ADD $s7, $s6, $s6 # double s6 by adding it to itself, should be 256
J CHANGE_V

CHANGE_T: ADD $t0, $s7, $s7
ADD $t1, $t0, $t0
ADD $t2, $t1, $t1
ADD $t3, $t2, $t2
ADD $t4, $t3, $t3
ADD $t5, $t4, $t4
ADD $t6, $t5, $t5
ADD $t7, $t6, $t6
ADD $t8, $t7, $t7
ADD $t9, $t8, $t8
J CHANGE_S

CHANGE_A: ADD $a0, $t9, $t9
ADD $a1, $a0, $a0
ADD $a2, $a1, $a1
ADD $a3, $a2, $a2
J CHANGE_S

CHANGE_V: ADD $v0, $a3, $a3
ADD $v1, $v0, $v0
J CHANGE_A
```

(Demo with `WeMIPS`)

In Pairs or Triples:

- Write a complete **Python program** that converts kilograms to pounds.
- *Predict what the C++ code will do:*

```cpp
//Another C++ program, demonstrating variables
#include <iostream>
using namespace std;

int main ()
{
  int year;
  cout << "Enter a number: ";
  cin >> year;
  cout << "Hello " << year << "!!\n\n";
  return 0;
}
```

Python Tutor

- Write a complete **Python program** that converts kilograms to pounds.

(Write from scratch in `pythonTutor`.)

onlinegdb demo

```cpp
//Another C++ program, demonstrating variables
#include <iostream>
using namespace std;

int main ()
{
  int year;
  cout << "Enter a number: ";
  cin >> year;
  cout << "Hello " << year << "!!\n\n";
  return 0;
}
```

(Demo with `onlinegdb`)

Introduction to C++

```
1  //Another C++ program, demonstrating variables
2  #include <iostream>
3  using namespace std;
4
5  int main ()
6  {
7    int year;
8    cout << "Enter a number: ";
9    cin >> year;
10   cout << "Hello " << year << "!!\n\n";
11   return 0;
12 }
```

- C++ is a popular programming language that extends C.

Introduction to C++

```cpp
//Another C++ program, demonstrating variables
#include <iostream>
using namespace std;

int main ()
{
  int year;
  cout << "Enter a number: ";
  cin >> year;
  cout << "Hello " << year << "!!\n\n";
  return 0;
}
```

- C++ is a popular programming language that extends C.
- Fast, efficient, and powerful.

Introduction to C++

```cpp
//Another C++ program, demonstrating variables
#include <iostream>
using namespace std;

int main ()
{
  int year;
  cout << "Enter a number: ";
  cin >> year;
  cout << "Hello " << year << "!!\n\n";
  return 0;
}
```

- C++ is a popular programming language that extends C.
- Fast, efficient, and powerful.
- Used for systems programming (and future courses!).

Introduction to C++

```cpp
//Another C++ program, demonstrating variables
#include <iostream>
using namespace std;

int main ()
{
  int year;
  cout << "Enter a number: ";
  cin >> year;
  cout << "Hello " << year << "!!\n\n";
  return 0;
}
```

- C++ is a popular programming language that extends C.
- Fast, efficient, and powerful.
- Used for systems programming (and future courses!).
- Today, we'll introduce the basic structure and simple input/output (I/O) in C/C++.

Introduction to C++

- Programs are organized in functions.

```
1  //Another C++ program, demonstrating variables
2  #include <iostream>
3  using namespace std;
4
5  int main ()
6  {
7    int year;
8    cout << "Enter a number: ";
9    cin >> year;
10   cout << "Hello " << year << "!!\n\n";
11   return 0;
12 }
```

Introduction to C++

- Programs are organized in functions.
- Variables must be **declared** before used:

```
1  //Another C++ program, demonstrating variables
2  #include <iostream>
3  using namespace std;
4
5  int main ()
6  {
7    int year;
8    cout << "Enter a number: ";
9    cin >> year;
10   cout << "Hello |" << year << "!!\n\n";
11   return 0;
12 }
```

Introduction to C++

- Programs are organized in functions.
- Variables must be **declared** before used:
  `int num;`

```
1  //Another C++ program, demonstrating variables
2  #include <iostream>
3  using namespace std;
4
5  int main ()
6  {
7    int year;
8    cout << "Enter a number: ";
9    cin >> year;
10   cout << "Hello " << year << "!!\n\n";
11   return 0;
12 }
```

Introduction to C++

```cpp
//Another C++ program, demonstrating variables
#include <iostream>
using namespace std;

int main ()
{
  int year;
  cout << "Enter a number: ";
  cin >> year;
  cout << "Hello " << year << "!!\n\n";
  return 0;
}
```

- Programs are organized in functions.
- Variables must be **declared** before used:
  `int num;`
- Many types available:
  `int, float, char, ...`

Introduction to C++

```cpp
//Another C++ program, demonstrating variables
#include <iostream>
using namespace std;

int main ()
{
  int year;
  cout << "Enter a number: ";
  cin >> year;
  cout << "Hello " << year << "!!\n\n";
  return 0;
}
```

- Programs are organized in functions.
- Variables must be **declared** before used:
  `int num;`
- Many types available:
  `int, float, char, ...`
- To print, we'll use `cout <<`:

Introduction to C++

```
1  //Another C++ program, demonstrating variables
2  #include <iostream>
3  using namespace std;
4
5  int main ()
6  {
7    int year;
8    cout << "Enter a number: ";
9    cin >> year;
10   cout << "Hello " << year << "!!\n\n";
11   return 0;
12 }
```

- Programs are organized in functions.
- Variables must be **declared** before used:
  `int num;`
- Many types available:
  `int, float, char, ...`
- To print, we'll use `cout <<`:
  `cout << "Hello!!"`

Introduction to C++

```
1  //Another C++ program, demonstrating variables
2  #include <iostream>
3  using namespace std;
4
5  int main ()
6  {
7    int year;
8    cout << "Enter a number: ";
9    cin >> year;
10   cout << "Hello |" << year << "!!\n\n";
11   return 0;
12 }
```

- Programs are organized in functions.
- Variables must be **declared** before used:
  `int num;`
- Many types available:
  `int, float, char, ...`
- To print, we'll use `cout <<`:
  `cout << "Hello!!"`
- To get input, we'll use `cin >>`:

Introduction to C++

```
1  //Another C++ program, demonstrating variables
2  #include <iostream>
3  using namespace std;
4
5  int main ()
6  {
7    int year;
8    cout << "Enter a number: ";
9    cin >> year;
10   cout << "Hello |" << year << "!!\n\n";
11   return 0;
12 }
```

- Programs are organized in functions.
- Variables must be **declared** before used:
  `int num;`
- Many types available:
  `int, float, char, ...`
- To print, we'll use `cout <<`:
  `cout << "Hello!!"`
- To get input, we'll use `cin >>`:
  `cin >> num`

Introduction to C++

```
1  //Another C++ program, demonstrating variables
2  #include <iostream>
3  using namespace std;
4
5  int main ()
6  {
7    int year;
8    cout << "Enter a number: ";
9    cin >> year;
10   cout << "Hello |" << year << "!!\n\n";
11   return 0;
12 }
```

- Programs are organized in functions.
- Variables must be **declared** before used:
  `int num;`
- Many types available:
  `int, float, char, ...`
- To print, we'll use `cout <<`:
  `cout << "Hello!!"`
- To get input, we'll use `cin >>`:
  `cin >> num`
- To use those I/O functions, we put at the top of the program:

Introduction to C++

```
1  //Another C++ program, demonstrating variables
2  #include <iostream>
3  using namespace std;
4
5  int main ()
6  {
7    int year;
8    cout << "Enter a number: ";
9    cin >> year;
10   cout << "Hello |" << year << "!!\n\n";
11   return 0;
12 }
```

- Programs are organized in functions.
- Variables must be **declared** before used:
  `int num;`
- Many types available:
  `int, float, char, ...`
- To print, we'll use `cout <<`:
  `cout << "Hello!!"`
- To get input, we'll use `cin >>`:
  `cin >> num`
- To use those I/O functions, we put at the top of the program:
  `#include <iostream>`
  `using namespace std;`

In Pairs or Triples:

Predict what the following pieces of code will do:

```
//Another C++ program, demonstrating I/O & arithmetic
#include <iostream>
using namespace std;

int main ()
{
  float kg, lbs;
  cout << "Enter kg: ";
  cin >> kg;
  lbs = kg * 2.2;
  cout << endl << "Lbs: " << lbs << "\n\n";
  return 0;
}
```

C++ Demo

```cpp
//Another C++ program, demonstrating I/O & arithmetic
#include <iostream>
using namespace std;

int main ()
{
  float kg, lbs;
  cout << "Enter kg: ";
  cin >> kg;
  lbs = kg * 2.2;
  cout << endl << "Lbs: " << lbs << "\n\n";
  return 0;
}
```

(Demo with `onlinegdb`)

In Pairs or Triples:

Predict what the following pieces of code will do:

```
//Another C++ program; Demonstrates loops
#include <iostream>
using namespace std;

int main ()
{
  int i,j;
  for (i = 0; i < 4; i++)
  {
      cout << "The world turned upside down...\n";
  }

  for (j = 10; j > 0; j--)
  {
      cout << j << " ";
  }
  cout << "Blast off!!" << endl;

  return 0;
}
```

C++ Demo

```cpp
//Another C++ program; Demonstrates loops
#include <iostream>
using namespace std;

int main ()
{
  int i,j;
  for (i = 0; i < 4; i++)
  {
     cout << "The world turned upside down...\n";
  }

  for (j = 10; j > 0; j--)
  {
     cout << j << " ";
  }
  cout << "Blast off!!" << endl;

  return 0;
}
```

(Demo with `onlinegdb`)

Definite loops

```
//Another C++ program; Demonstrates loops
#include <iostream>
using namespace std;

int main ()
{
  int i,j;
  for (i = 0; i < 4; i++)
  {
      cout << "The world turned upside down...\n";
  }

  for (j = 10; j > 0; j--)
  {
      cout << j << " ";
  }
  cout << "Blast off!!" << endl;

  return 0;
}
```

General format:

`for` (*initialization* ; *test* ; *updateAction*)
{

command1;
command2;
command3;
...

}

In Pairs or Triples:

Predict what the following pieces of code will do:

```
//Growth example
#include <iostream>
using namespace std;

int main ()
{
  int population = 100;
  cout << "Year\tPopulation\n";
  for (int year = 0; year < 100; year= year+5)
  {
      cout << year << "\t" << population << "\n";
      population = population * 2;

  }
  return 0;
}
```

C++ Demo

```cpp
//Growth example
#include <iostream>
using namespace std;

int main ()
{
  int population = 100;
  cout << "Year\tPopulation\n";
  for (int year = 0; year < 100; year= year+5)
  {
      cout << year << "\t" << population << "\n";
      population = population * 2;

  }
  return 0;
}
```

(Demo with C++)

In Pairs or Triples:

Predict what the following pieces of code will do:

```
//Another C++ program; Demonstrates loops
#include <iostream>
using namespace std;

int main ()
{
  int i,j,size;
  cout << "Enter size: ";
  cin >> size;
  for (i = 0; i < size; i++)
  {
    for (j = 0; j < size; j++)
      cout << "*";
    cout << endl;
  }
  cout << "\n\n";
  for (i = size; i > 0; i--)
  {
    for (j = 0; j < i; j++)
      cout << "*";
    cout << endl;
  }
  return 0;
}
```

C++ Demo

```cpp
//Another C++ program; Demonstrates loops
#include <iostream>
using namespace std;

int main ()
{
  int i,j,size;
  cout << "Enter size: ";
  cin >> size;
  for (i = 0; i < size; i++)
  {
    for (j = 0; j < size; j++)
      cout << "*";
    cout << endl;
  }
  cout << "\n\n";
  for (i = size; i > 0; i--)
  {
    for (j = 0; j < i; j++)
      cout << "*";
    cout << endl;
  }
  return 0;
}
```

(Demo with C++)

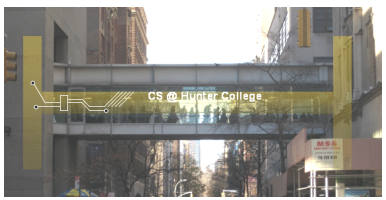
Lecture Slips

In pairs or triples: **translate** the C++ program into Python:

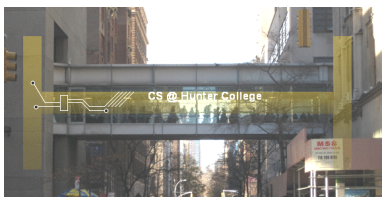
```cpp
//Growth example
#include <iostream>
using namespace std;

int main ()
{
  int population = 100;
  cout << "Year\tPopulation\n";
  for (int year = 0; year < 100; year= year+5)
  {
      cout << year << "\t" << population << "\n";
      population = population * 2;

  }
  return 0;
}
```

Recap: C++

- On lecture slip, write down a topic you wish we had spent more time (and why).

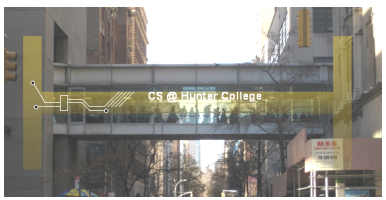


Recap: C++



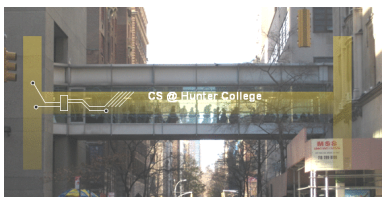
- On lecture slip, write down a topic you wish we had spent more time (and why).
- C++ is a popular programming language that extends C.

Recap: C++



- On lecture slip, write down a topic you wish we had spent more time (and why).
- C++ is a popular programming language that extends C.
- Input/Output (I/O):
  - `cin >>`
  - `cout <<`

Recap: C++



- On lecture slip, write down a topic you wish we had spent more time (and why).
- C++ is a popular programming language that extends C.
- Input/Output (I/O):
  - `cin >>`
  - `cout <<`
- Definite loops:
  `for (i = 0; i < 10; i++)`

Practice Quiz & Final Questions



- Lightning rounds:

Practice Quiz & Final Questions



- Lightning rounds:
  - write as much you can for 60 seconds;

Practice Quiz & Final Questions



- Lightning rounds:
  - write as much you can for 60 seconds;
  - followed by answer; and

Practice Quiz & Final Questions



- Lightning rounds:
  - write as much you can for 60 seconds;
  - followed by answer; and
  - repeat.

Practice Quiz & Final Questions



- Lightning rounds:
  - write as much you can for 60 seconds;
  - followed by answer; and
  - repeat.
- Continue from last time on the mock exam (on web page).