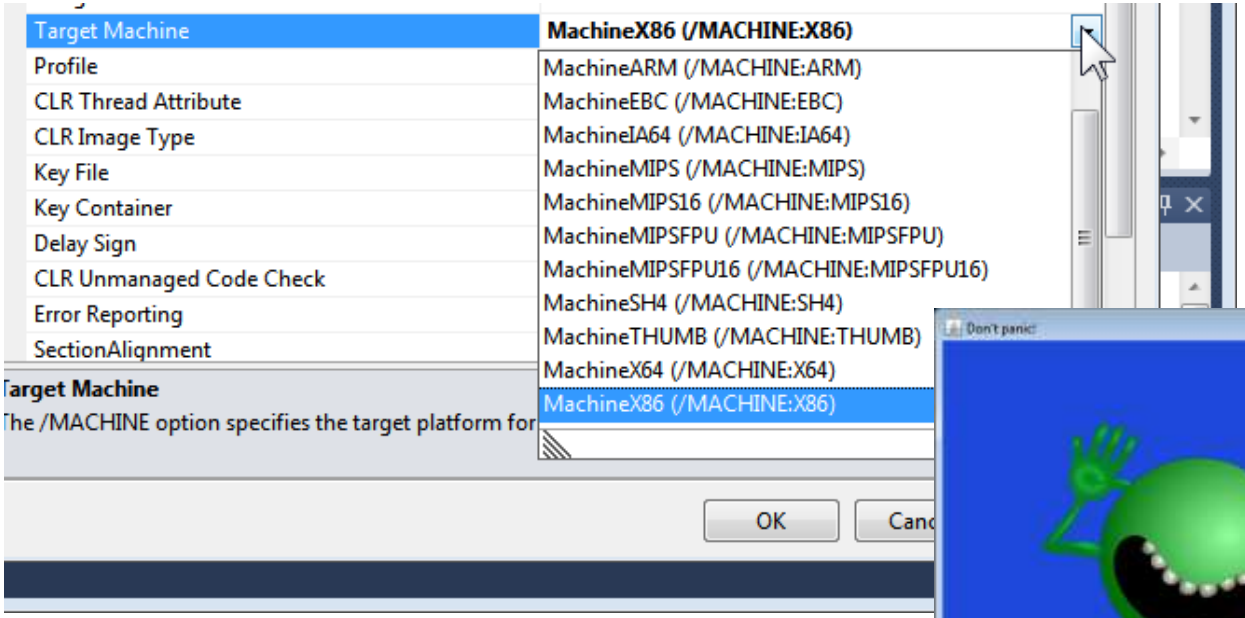
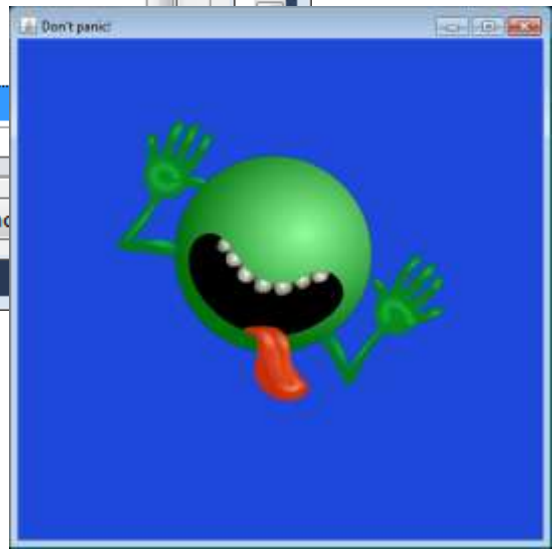
Application Deployment





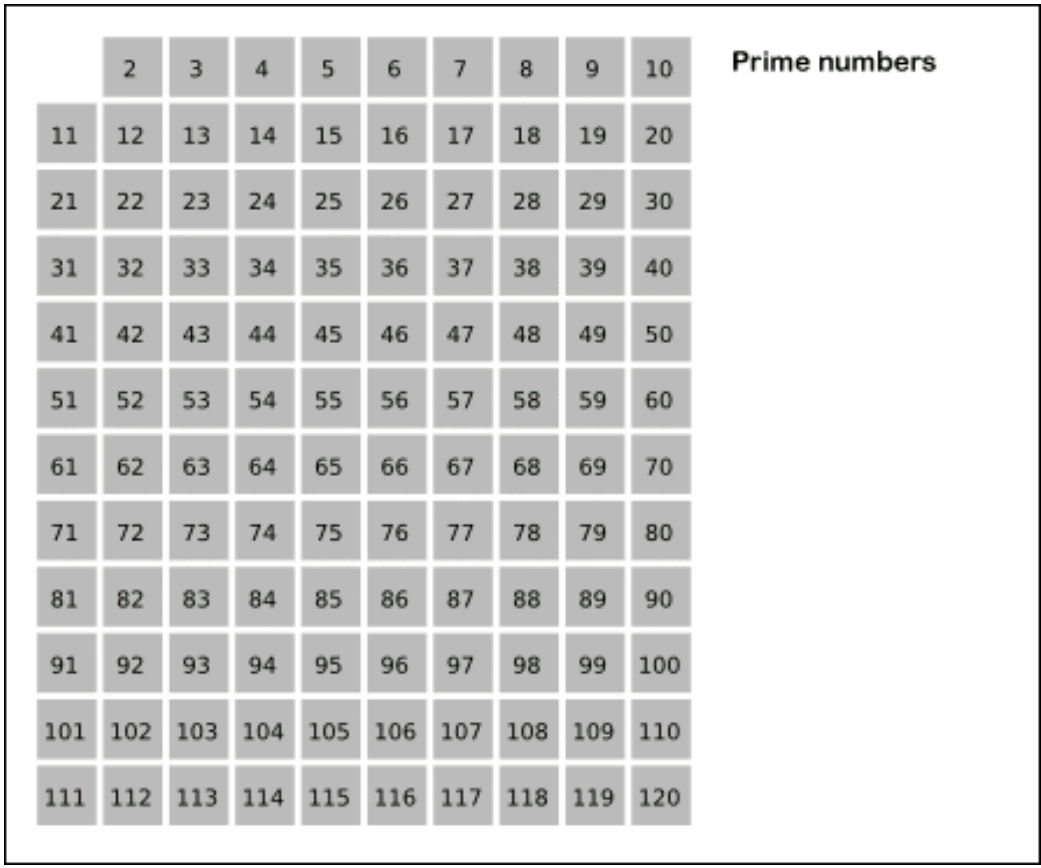
Overview

- Platform (in)dependence
  - Native compilation
    - e.g. C++
  - Bytecode
    - e.g. Java, .NET
  - Interpreted
    - e.g. PHP, JavaScript
- Performance comparison

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main(int argc, char** argv)
{
   int n = atoi(argv[1]);

   bool* a = new bool[n + 1];
   for (int i = 0; i <= n; i++)
   a[i] = true;

   int top = (int) sqrt((double) n);
   for (int i = 2; i <= top; i++)
   {
      int j = 2 * i;
      while (j <= n)
      {
         a[j] = false;
         j += i;
      }
   }

   for (int i = 2; i <= n; i++)
      if (a[i]) printf("%d ", i);
   printf("\n");
   delete [] a;
   return 0;
}
```

Primes.cpp

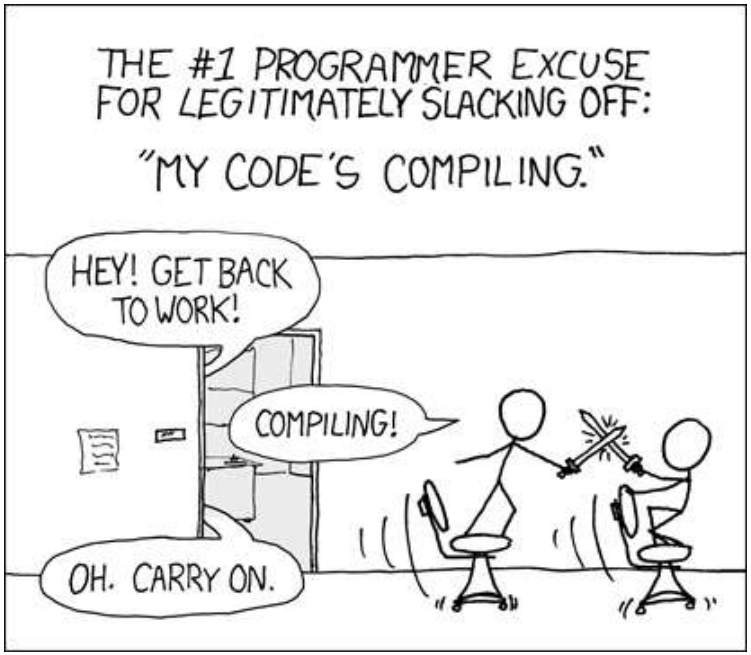
Sieve of Eratosthenes



	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100
101	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120

Native compilation

- C/C++
  - Write the code
  - Press the compile button
  - Wait
  - Fix compile errors
  - Press the compile button
  - …
  - Native code loaded directly on CPU
  - Code is specific to the targeted architecture
    - Compile for Windows x86
    - Compile for Mac x86
    - …



http://xkcd.com/303/

Source code:

Plain text file created in a high-level programming language

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main(int argc, char** argv)
{
   int n = atoi(argv[1]);
   ...
```

Primes.cpp

Compile with GNU C++ compiler:

```
c:\source\c\Primes>g++ -O3 Primes.cpp -o Primes
```

Machine language:

Actual binary run by a particular processor, not human readable/writeable

Primes.exe

```
c:\source\c\Primes>Primes 100
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47
53 59 61 67 71 73 79 83 89 97
```

Specific to the architecture targeted by compiler (e.g. x86)

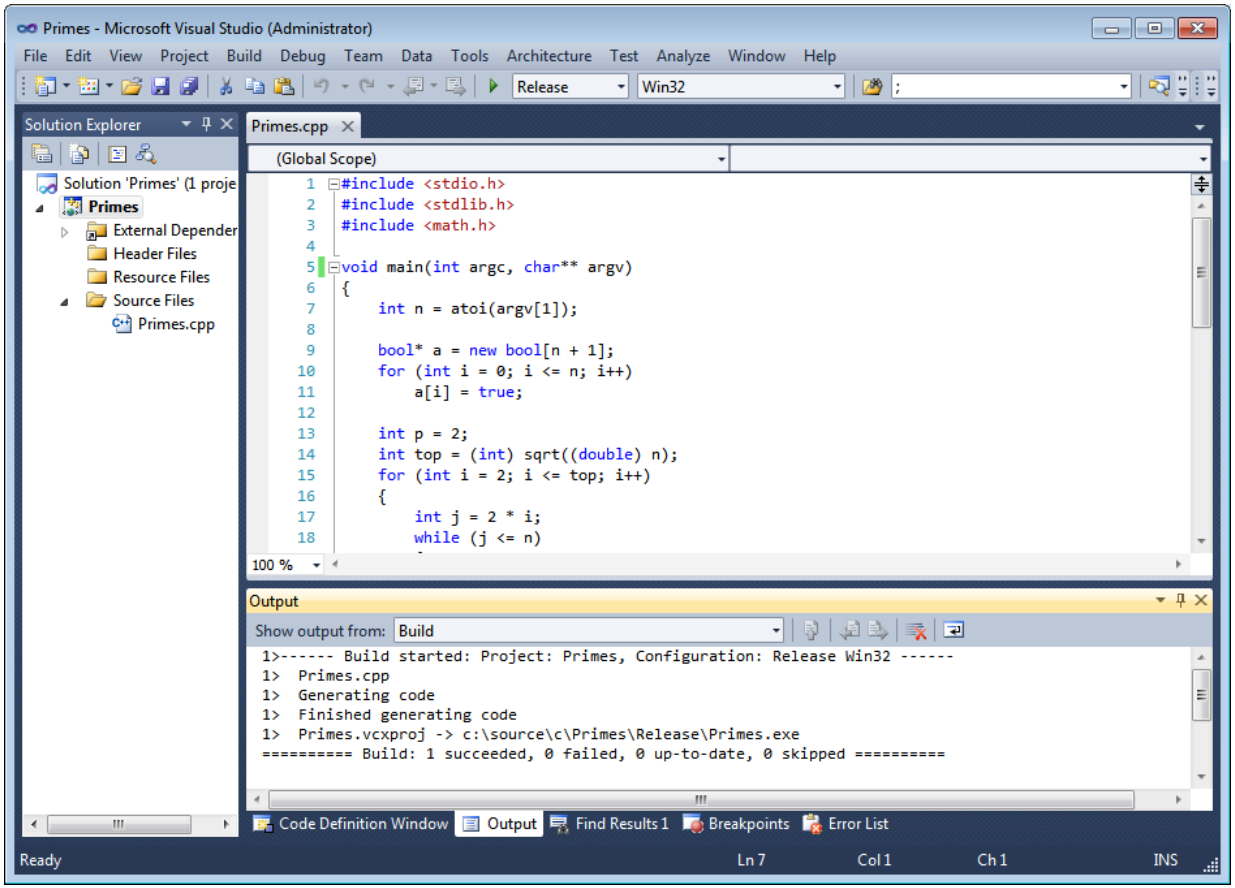
Source code:

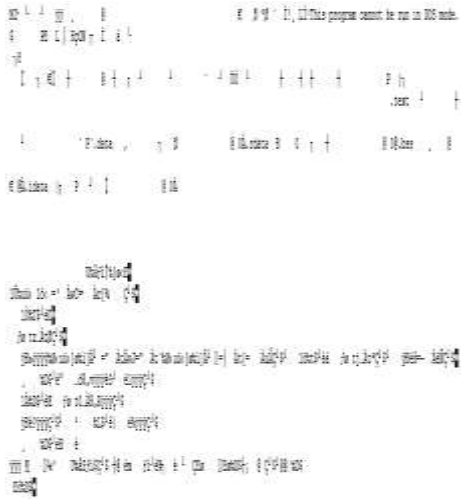
Plain text file created in a high-level programming language

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main(int argc, char** argv)
{
    int n = atoi(argv[1]);
    ...
```

Primes.cpp

Compile with Visual Studio:

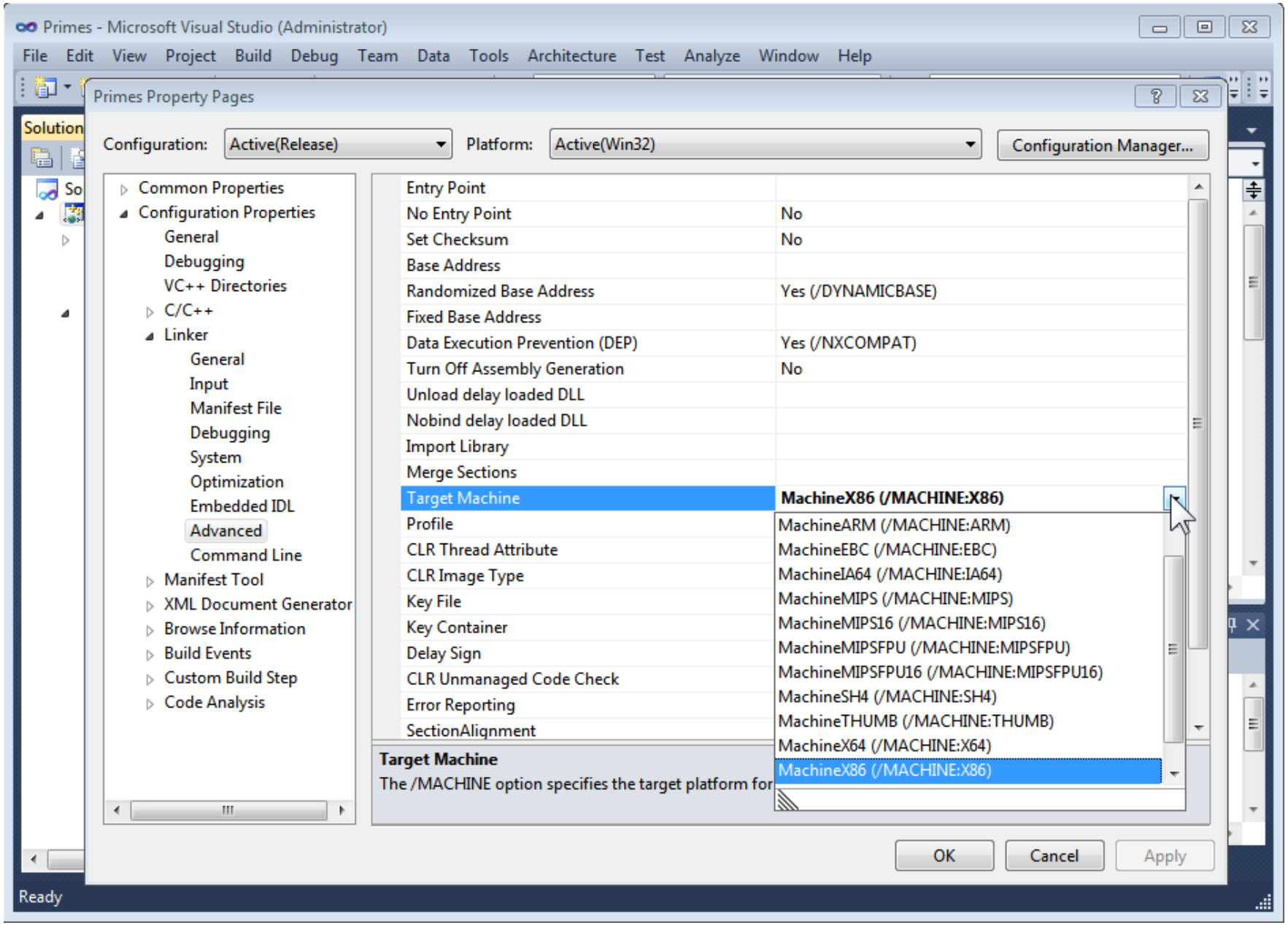




Primes.exe

Somewhat different binary than GNU's output, but should work on same architecture.

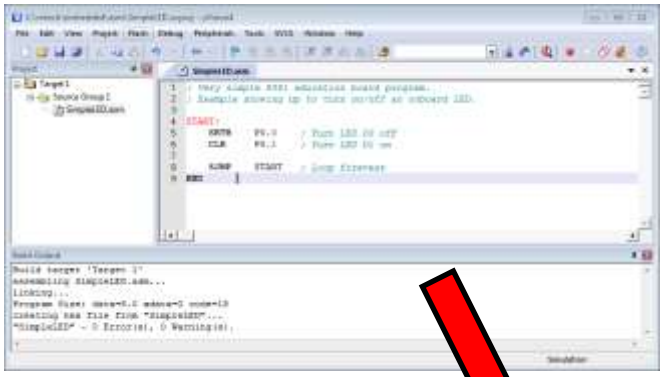
Different targets in Visual Studio



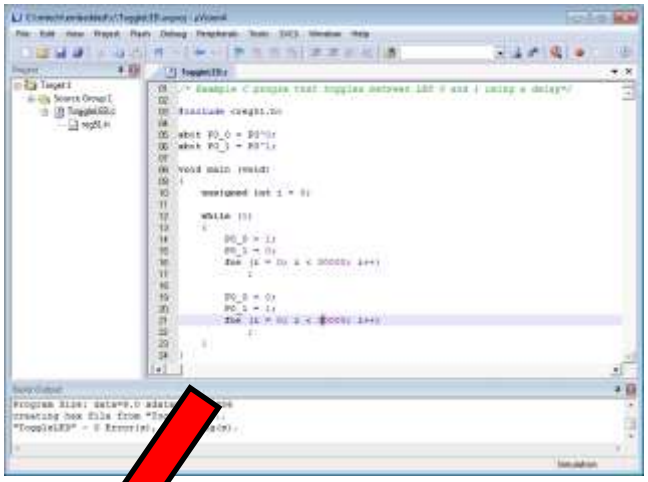
Assembly language

C

Keil IDE



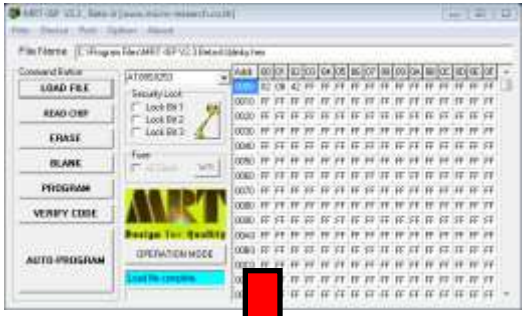
Primes.asm



Keil IDE

Primes.c

Hex file



MRT:
Loads machine code into code memory of 8051 board

8051 educational board

```java
public class Primes
{
   public static void main(String [] args)
   {
      int n = Integer.parseInt(args[0]);

      boolean [] a = new boolean[n + 1];
      for (int i = 0; i <= n; i++)
         a[i] = true;

      int top = (int) Math.sqrt((double) n);
      for (int i = 2; i <= top; i++)
      {
         int j = 2 * i;
         while (j <= n)
         {
            a[j] = false;
            j += i;
         }
      }

      for (int i = 2; i <= n; i++)
         if (a[i]) System.out.printf("%d ", i);
      System.out.printf("\n");
   }
}
```

Primes.java

```cpp
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main(int argc, char** argv)
{
   int n = atoi(argv[1]);

   bool* a = new bool[n + 1];
   for (int i = 0; i <= n; i++)
   a[i] = true;

   int top = (int) sqrt((double) n);
   for (int i = 2; i <= top; i++)
   {
      int j = 2 * i;
      while (j <= n)
      {
         a[j] = false;
         j += i;
      }
   }

   for (int i = 2; i <= n; i++)
      if (a[i]) printf("%d ", i);
   printf("\n");

   delete [] a;
   return 0;
}
```

Primes.cpp

Source code:

Plain text file created in some editor (notepad, vi, TextEdit, Eclipse, DrJava, …)

```
public class Primes
{
   public static void main(String [] args)
   {
      int n = Integer.parseInt(args[0]);
      ...
```

Primes.java

```
% javac Primes.java
```

Java bytecode:

Intermediate language that any device running Java can understand

```
☼LineNumberTable☺
♦main☺
—([Ljava/lang/String;)V☺
StackMapTable
#☺

SourceFile☺
♂Primes.java♀
♂
♀
*♀
+
,☺
♥%d ☺
►java/lang/Object♀
```

Primes.class

This guy will run on anything with a Java Virtual Machine (JVM)

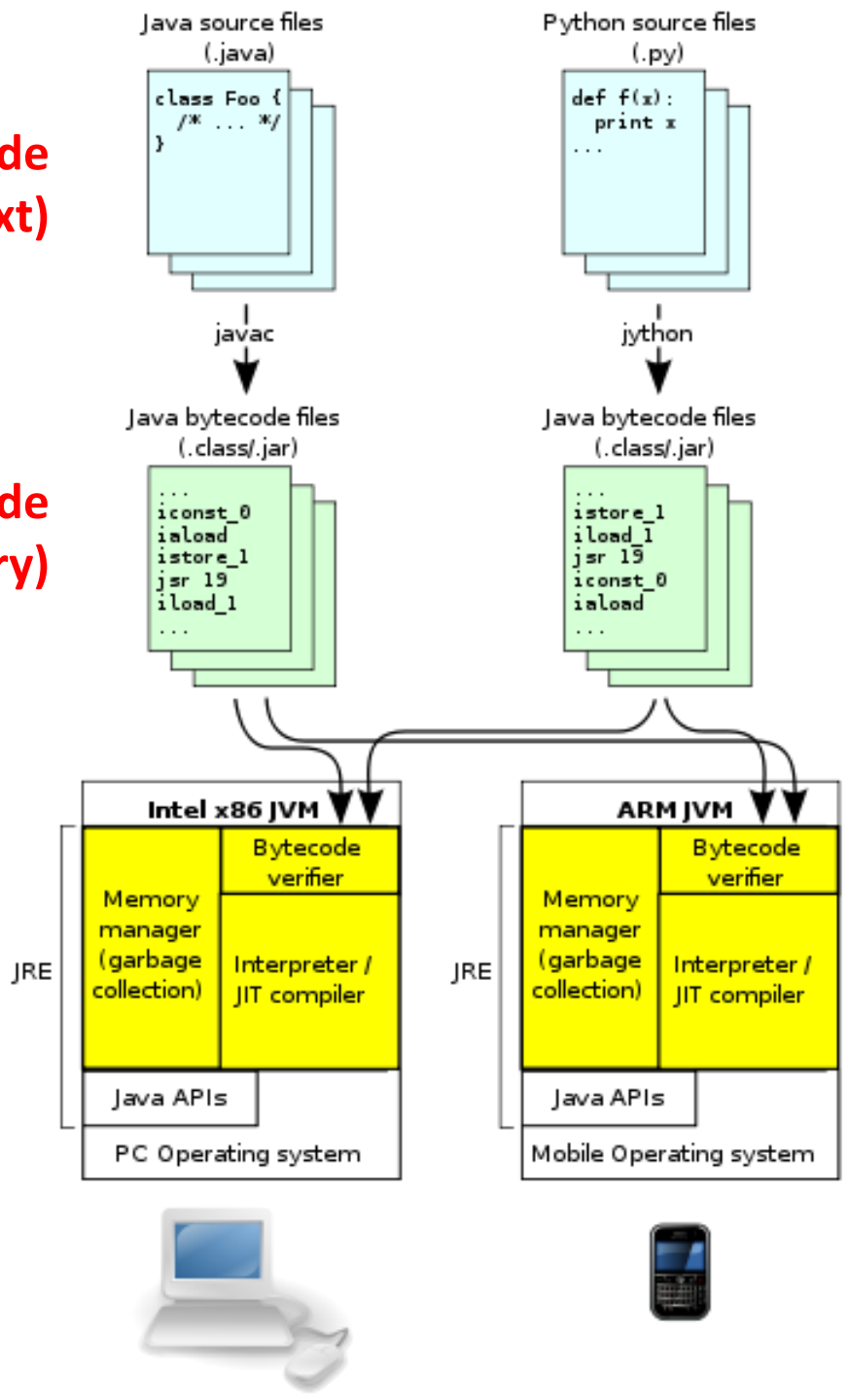
```
c:\source\c\Primes>java Primes 100
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47
53 59 61 67 71 73 79 83 89 97
```

Source code (text)

Platform independent bytecode (binary)

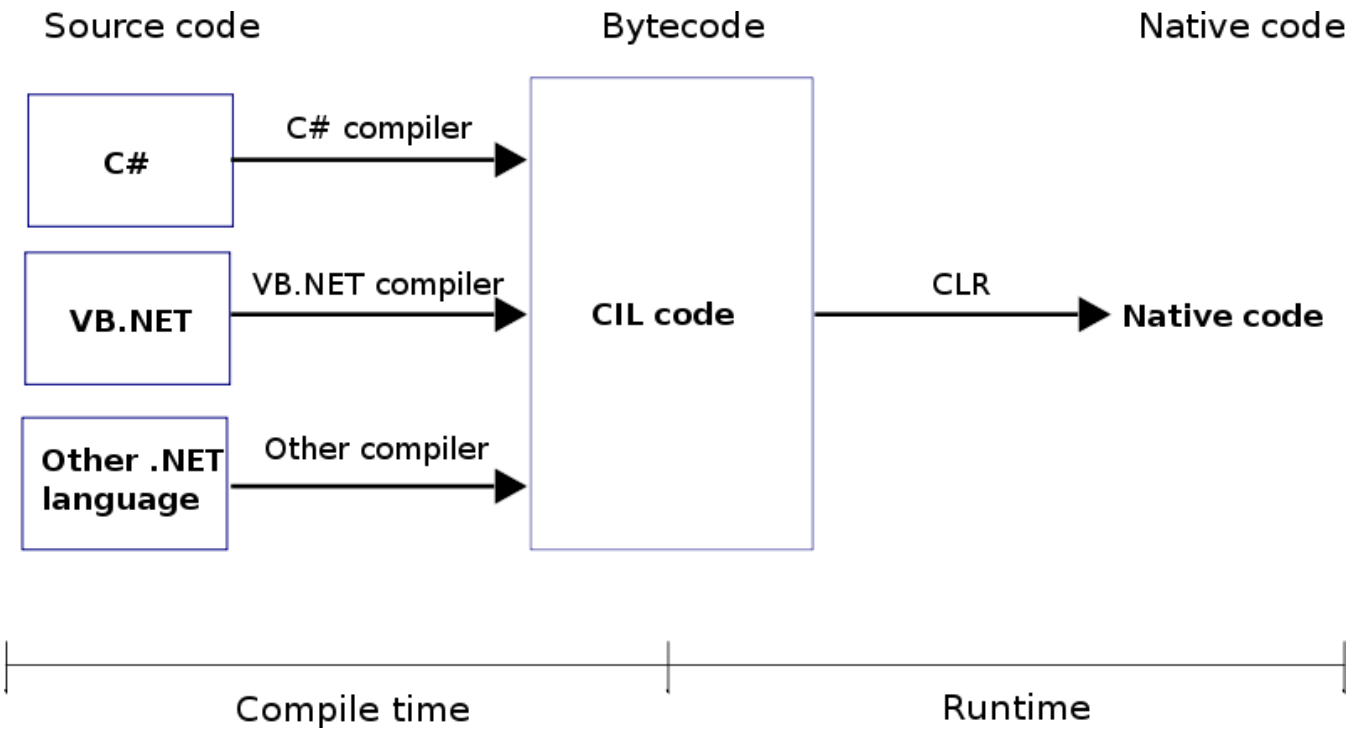
Platform dependent virtual machine (binary executable)

Target hardware



Other intermediate platforms

- Microsoft .NET framework
  - Write source code in a variety of languages:
    - C#, VB.NET, J#, F#
  - Compiled to Common Intermediate Language (CIL)
  - Bytecode executed by Common Language Runtime (CLR)



Mono

www.mono-project.com/Main_Page

mono

Search Mono

home download start news contribute community ios android support

Cross platform, open source .NET development framework

Mono

An open source, cross-platform, implementation of C# and the CLR that is binary compatible with Microsoft.NET

Learn More Download

MonoTouch for iOS

Build apps for iPhone and iPad using C#, MonoDevelop, and the Mono Framework

Learn More

Mono for Android

Build apps for Android devices using C#, Visual Studio or MonoDevelop, and the Mono Framework

Learn More

Run your applications on all the platforms

Mono is a software platform designed to allow developers to easily create cross platform applications. Sponsored by Xamarin, Mono is an open source implementation of Microsoft's .NET Framework based on the ECMA standards for C# and the Common Language Runtime. A growing family of solutions and an active and enthusiastic contributing community is helping position Mono to become the leading choice for development of Linux applications.

```cs
using System;

namespace Primes
{
   class Primes
   {
      static void Main(string[] args)
      {
         int n = Int32.Parse(args[0]);

         bool[] a = new bool[n + 1];
         for (int i = 0; i <= n; i++)
            a[i] = true;

         int top = (int) Math.Sqrt((double) n);
         for (int i = 2; i <= top; i++)
         {
            int j = 2 * i;
            while (j <= n)
            {
               a[j] = false;
               j += i;
            }
         }

         for (int i = 2; i <= n; i++)
            if (a[i])
               System.Console.Write(i + " ");
          System.Console.WriteLine();
      }
   }
}
```

Primes.cs

```java
public class Primes
{
   public static void main(String [] args)
   {
      int n = Integer.parseInt(args[0]);

      boolean [] a = new boolean[n + 1];
      for (int i = 0; i <= n; i++)
         a[i] = true;

      int top = (int) Math.sqrt((double) n);
      for (int i = 2; i <= top; i++)
      {
         int j = 2 * i;
         while (j <= n)
         {
            a[j] = false;
            j += i;
         }
      }

      for (int i = 2; i <= n; i++)
         if (a[i])
            System.out.printf("%d ", i);
      System.out.printf("\n");
   }
}
```

Primes.java

Bytecode advantages

- Platform independence
  - Write once, run anywhere
    - At anywhere someone has written a virtual machine
- Possibly faster than pre-compiling?
  - JIT compiler can optimize bytecode to CPU it is running on
    - Difficult to do this if you compile in advance
    - You would have to deploy a version for every CPU variant
- Security
  - Virtual Machine (VM) verify code is behaving itself

Interpreted languages

- No compilation step before running
- Each line compiled as encountered
  - Originally some programming languages
    - e.g. Smalltalk, BASIC
  - Other scripting languages
    - e.g. DOS batch files, bash scripts, PHP, JavaScript
  - Now most utilize Just In Time (JIT) compilers
    - Cache previously compiled lines of code
  - Many languages though of as "interpreted" use bytecode like approach
    - e.g. Python, Perl, Ruby

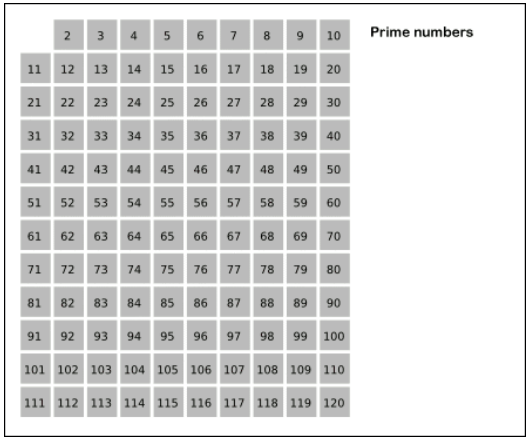
```php
<?php



$n = $_GET["N"];

$a = array();
for ($i = 0; $i <= $n; $i++)
    $a[$i] = true;

$top = intval(sqrt($n));
for ($i = 2; $i <= $top; $i++)
{
    $j = 2 * $i;
    while ($j <= $n)
    {
        $a[$j] = false;
        $j += $i;
    }
    for ($i = 2; $i <= $n; $i++)
       if ($a[$i]) echo "$i ";
    echo "<br />";
}

?>
```

Primes.php

```java
public class Primes
{
   public static void main(String [] args)
   {
      int n = Integer.parseInt(args[0]);

      boolean [] a = new boolean[n + 1];
      for (int i = 0; i <= n; i++)
         a[i] = true;

      int top = (int) Math.sqrt((double) n);
      for (int i = 2; i <= top; i++)
      {
         int j = 2 * i;
         while (j <= n)
         {
            a[j] = false;
            j += i;
         }
      }

      for (int i = 2; i <= n; i++)
         if (a[i])
            System.out.printf("%d ", i);
      System.out.printf("\n");
   }
}
```

Primes.java

Benchmarks



- Find the primes from 2 to 123,456,789
  - Using standard Sieve of Eratosthenes
  - Repeated ten times
  - Disabled console output for timing

Language	Compiler	Type	Tower	MacBook Pro	ThinkPad x41
C++	Visual Studio 2010	compiled	164.9	-	834.3
C++	GNU g++	compiled	161.0	75.0	830.7
Java	javac 1.6.0_26	bytecode	158.2	75.2	843.0
C#	Visual Studio 2010	bytecode	162.6	76.4	819.6
C#	Mono	bytecode	162.5	76.8	795.1
PHP	(only to 12,345,678) 1 GB memory	interpreted	300.9	258.1	568.8

Summary

- Deploying your program
  - Compiled to native code
  - Compiled to byte code
    - Before you deploy
    - Just-in-time based on source
  - Interpreted
- Benchmark results
  - Native and intermediate byte code, similar
  - Interpreted much slower