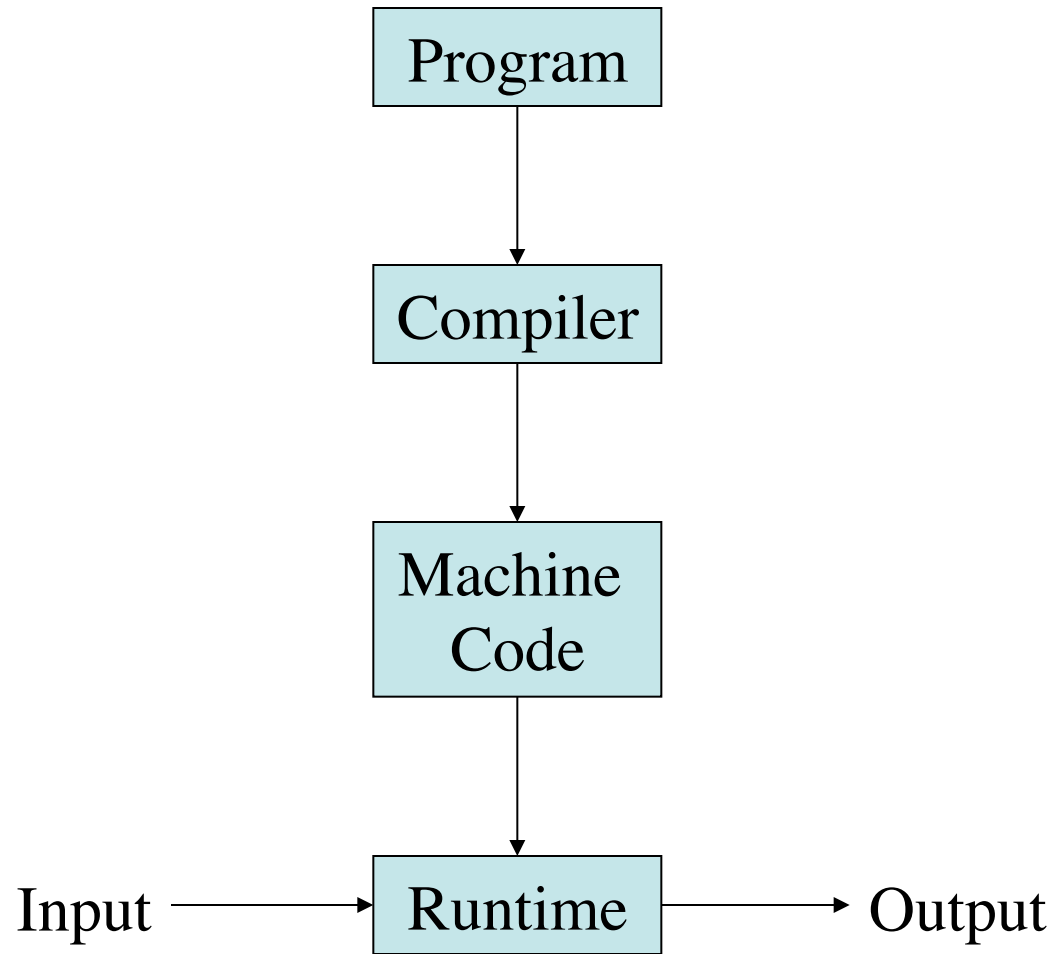


CMPT 379

Compilers

Anoop Sarkar

<http://www.cs.sfu.ca/~anoop>



Compilers

- Analysis of the source (front-end)
- Synthesis of the target (back-end)
- The *translation* from user **intention** into intended **meaning**
- The requirements from a Compiler and a Programming Language are:
 - Ease of use (high-level programming)
 - Speed

Cousins of the compiler

- “Smart” editors for structured languages
 - static checkers; pretty printers
- Structured or semi-structured data
 - Trees as data: s-expressions; XML
 - query languages for databases: SQL
- Interpreters (for PLs like lisp or scheme)
 - Scripting languages: perl, python, tcl/tk
 - Special scripting languages for applications
 - “Little” languages: awk, eqn, troff, TeX
- Compiling to Bytecode (virtual machines)

Program



Compiler

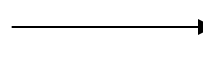


Machine
Code

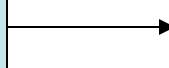


Runtime

Input



Output



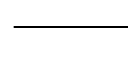
Static

Program

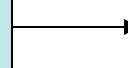


Interpreter

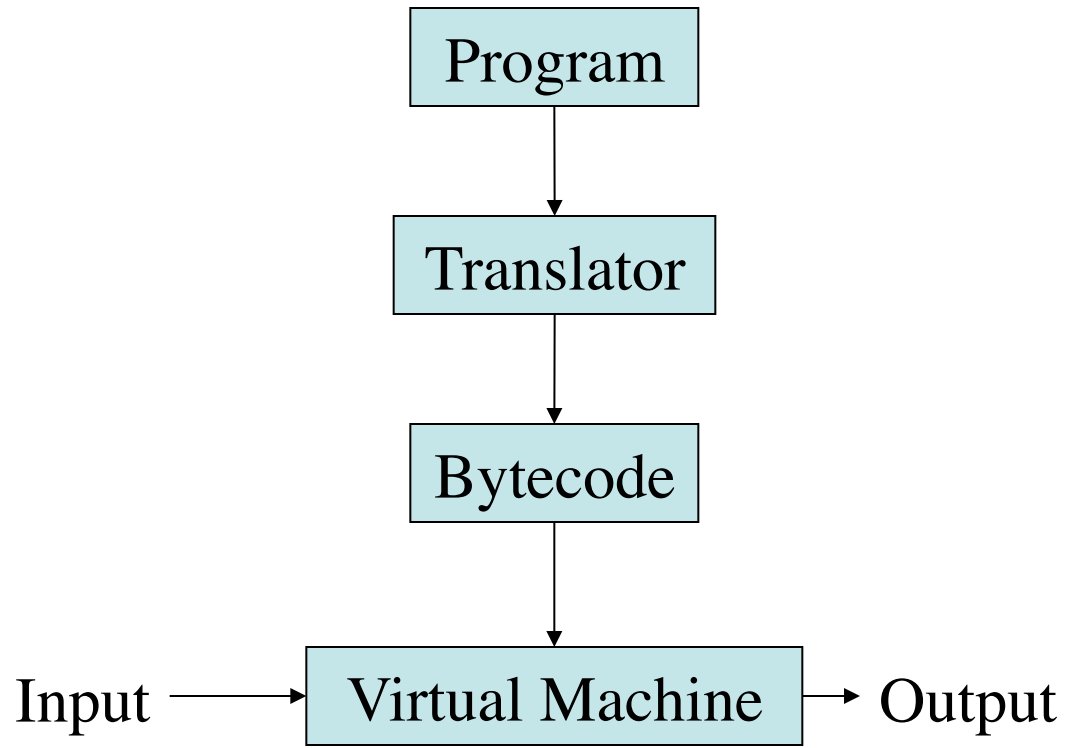
Input



Output



Dynamic



Static/Dynamic

Context for the Compiler

- Preprocessor
- Compiler
- Assembler
- Linker (loader)

MIPS CPU

Program Counter

PC = 00000000 EPC = 00000000 Cause = 00000000 BadVaddr = 00000000
 Status = 00000000 HI = 00000000 LO = 00000000

General registers

R0 (r0) = 00000000 R8 (t0) = 00000000 R24 (t8) = 00000000
 R1 (at) = 00000000 R9 (t1) = 00000000 R25 (s9) = 00000000
 R2 (v0) = 00000000 R10 (t2) = 00000000 R26 (k0) = 00000000
 R3 (v1) = 00000000 R11 (t3) = 00000000 R27 (k1) = 00000000
 R4 (a0) = 00000000 R12 (t4) = 00000000 R28 (gp) = 00000000
 R5 (a1) = 00000000 R13 (t5) = 00000000 R29 (sp) = 00000000
 R6 (a2) = 00000000 R14 (t6) = 00000000 R30 (s8) = 00000000
 R7 (a3) = 00000000 R15 (t7) = 00000000 R23 (s7) = 00000000 R31 (ra) = 00000000

\$a0 to \$a3 used to pass arguments to a function call

Double floating-point registers

FP0 = 0.000000 FP8 = 0.000000 FP16 = 0.000000 FP24 = 0.000000
 FP2 = 0.000000 FP10 = 0.000000 FP18 = 0.000000 FP26 = 0.000000
 FP4 = 0.000000 FP12 = 0.000000 FP20 = 0.000000 FP28 = 0.000000
 FP6 = 0.000000 FP14 = 0.000000 FP22 = 0.000000 FP30 = 0.000000

Single floating-point registers

MIPS CPU

Text segments

[0x00400000]	0x8fa40000	lw \$4, 0(\$29)	; 89: lw \$a0, 0(\$sp)
[0x00400004]	0x27a50004	addiu \$5, \$29, 4	; 90: addiu \$a1, \$sp, 4
[0x00400008]	0x24a60004	addiu \$6, \$5, 4	; 91: addiu \$a2, \$a1, 4
[0x0040000c]	0x00041080	sll \$2, \$4, 2	; 92: sll \$v0, \$a0, 2
[0x00400010]	0x00c23021	addu \$6, \$6, \$2	; 93: addu \$a2, \$a2, \$v0
[0x00400014]	0x0c000000	jal 0x00000000 [main]	; 94: jal main
[0x00400018]	0x3402000a	ori \$2, \$0, 10	; 95: li \$v0 10
[0x0040001c]	0x0000000c	syscall	; 96: syscall

Data segments

[0x10000000]	...	[0x10010000]	0x00000000	
[0x10010004]	0x74706563	0x206e6f69	0x636f2000	
[0x10010010]	0x72727563	0x61206465	0x6920646e	0x726f6e67
[0x10010020]	0x000a6465	0x495b2020	0x7265746e	0x74707572
[0x10010030]	0x0000205d	0x20200000	0x616e555b	0x6e67696c
[0x10010040]	0x61206465	0x65726464	0x69207373	0x6e69206e
[0x10010050]	0x642f7473	0x20617461	0x63746566	0x00205d68
[0x10010060]	0x555b2020	0x696c616e	0x64656e67	0x64646120
[0x10010070]	0x73736572	0x206e6920	0x726f7473	0x00205d65

What we understand

```
#include <stdio.h>

int main (int argc, char *argv[]) {
    int i;
    int sum = 0;
    for (i = 0; i <= 100; i++)
        sum = sum + i * i;
    printf ("Sum from 0..100 = %d\n", sum);
}
```

Assembly language

```
.text
.align 2
.globl main
main:
    subu $sp, $sp, 32
    sw $ra, 20($sp)
    sd $a0, 32($sp)
    sw $0, 24($sp)
    sw $0, 28($sp)
loop:
    lw $t6, 28($sp)
    mul $t7, $t6, $t6
    lw $t8, 24($sp)
    addu $t9, $t8, $t7
    sw $t9, 24($sp)
    addu $t0, $t6, 1
    sw $t0, 28($sp)
    ble $t0, 100, loop
    la $a0, str
    lw $a1, 24($sp)
    jal printf
    move $v0, $0
    lw $ra, 20($sp)
    addu $sp, $sp, 32
    jr $ra
.data
.align 0
str:
.asciiz "The sum from 0 .. 100 is %d\n"
```

A one-one translation from assembly to machine code

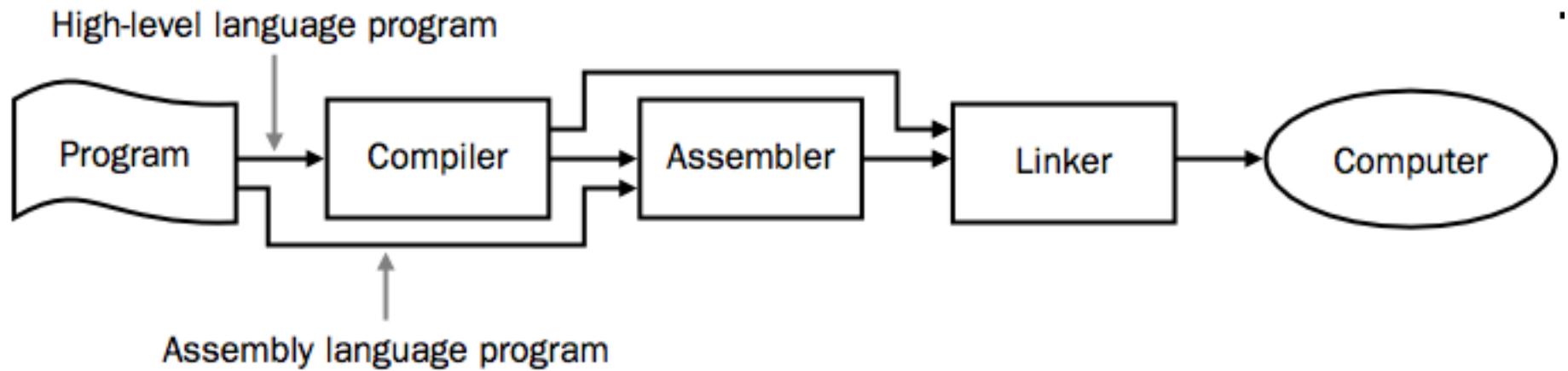
```

001001111011110111111111111111100000
1010111110111111100000000000010100
101011111010010000000000000100000
101011111010010100000000000100100
10101111101000000000000000011000
10101111101000000000000000011100
10001111101011100000000000011100
10001111101110000000000000011000
00000001110011100000000000011001
001001011100100000000000000000001
00101001000000010000000001100101
10101111101010000000000000011100
000000000000000000111100000010010
00000011000011111100100000100001
000101000010000011111111111110111
10101111101110010000000000011000
00111100000001000001000000000000
10001111101001010000000000011000
000011000001000000000000011101100
00100100100001000000010000110000
10001111101111110000000000010100
00100111101111010000000000100000
0000001111100000000000000001000
000000000000000000001000000100001

```

Conversion into instructions for the Machine

MIPS
machine language
code



Linker

.data

str:

.asciiz "the answer = "

.text

main:

li \$v0, 4

la \$a0, str

syscall

li \$v0, 1

li \$a0, 42

syscall

Local vs. Global labels

2-pass assembler and Linker

The UNIX toolchain (as, ar, ranlib, ld, ...)

