CS Teaching Academy

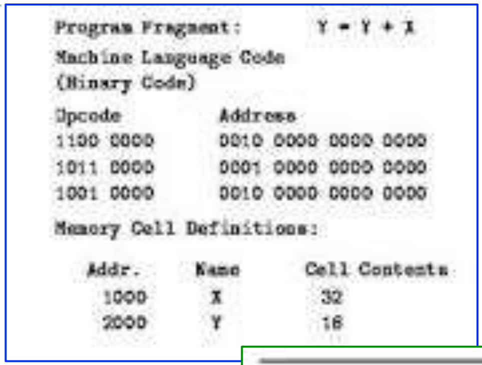
Chapter 6, Module 1: Programming Language Implementation

Objectives

- Identify the essential phases of program compilation
- Understand the different modes of language implementation

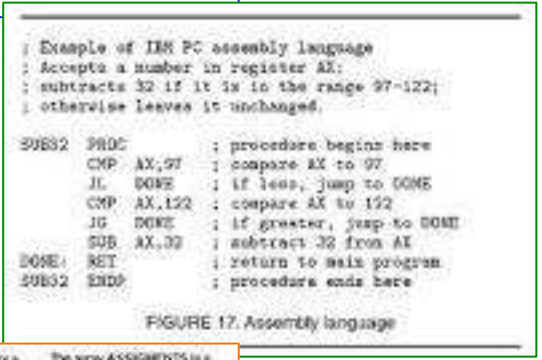
Language Generations

- Gen 1: Machine Language



```
Program Fragment:          Y = Y + X
Machine Language Code
(Binary Code)
Opcode          Address
1100 0000       0010 0000 0000 0000
1011 0000       0001 0000 0000 0000
1001 0000       0010 0000 0000 0000
Memory Cell Definitions:
   Addr.      Name       Cell Contents
    1000       X           32
    2000       Y           16
```

- Gen 2: Assembly Language
  - mnemonics vs. binary

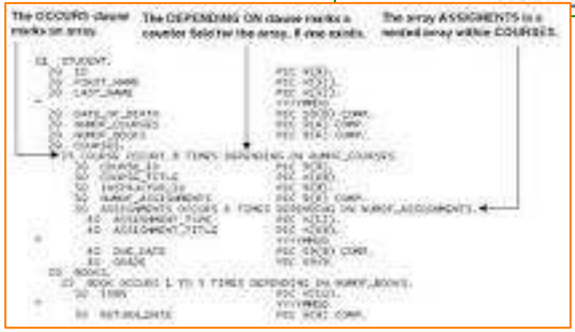


```
; Example of IBM PC assembly language
; Accepts a number in register AX;
; subtracts 32 if it is in the range 97-122;
; otherwise leaves it unchanged.

SUB32  PROC            ; procedure begins here
       CMP  AX,97      ; compare AX to 97
       JL   DONE       ; if less, jump to DONE
       CMP  AX,122     ; compare AX to 122
       JG   DONE       ; if greater, jump to DONE
       SUB  AX,32      ; subtract 32 from AX
DONE:  RET             ; return to main program
SUB32  ENDP            ; procedure ends here
```

FIGURE 17. Assembly language

- Gen 3: High-Level Language (e.g., Fortran, Cobol, ...)
  - powerful expressions



The OCCURS clause marks an array.

The DEPENDING ON clause marks a counter field for the array, if one exists.

The array ASSIGNMENTS is a nested array within COURSES.

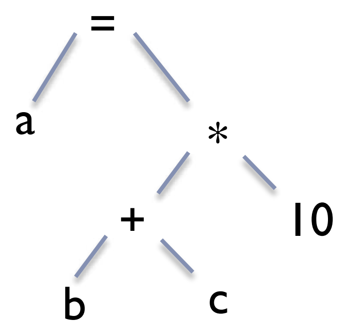
```
01  STUDENT.
    05  ID                                  PIC X(9).
    05  FIRST_NAME                          PIC X(12).
    05  LAST_NAME                           PIC X(12).
*                                           YYYYMMDD
    05  DATE_OF_BIRTH                       PIC S9(8) COMP.
    05  NUMBR_COURSES                       PIC 9(4) COMP.
    05  NUMBR_BOOKS                         PIC 9(4) COMP.
    05  COURSES.
     10 COURSE OCCURS 8 TIMES DEPENDING ON NUMBR_COURSES.
       20  COURSE_ID                        PIC 9(8).
       20  COURSE_TITLE                     PIC X(30).
       20  INSTRUCTOR_ID                    PIC 9(8).
       20  NUMBR_ASSIGNMENTS                PIC 9(4) COMP.
       20  ASSIGNMENTS OCCURS 8 TIMES DEPENDING ON NUMBR_ASSIGNMENTS.
         40  ASSIGNMENT_TYPE                PIC X(12).
         40  ASSIGNMENT_TITLE               PIC X(30).
*                                           YYYYMMDD
         40  DUE_DATE                       PIC S9(8) COMP.
         40  GRADE                          PIC 9(3).
    05  BOOKS.
     10 BOOK OCCURS 1 TO 9 TIMES DEPENDING ON NUMBR_BOOKS.
       20  ISBN                             PIC X(10).
*                                           YYYYMMDD
       20  RETURN_DATE                      PIC 9(8) COMP.
```

- Gen 4: ??? report generators, visual programming

Program Compilation Phases

1. Lexical Analysis
   - What elements are in the program?
     `a = b + c * 10 ;`

2. Parsing
   - What does the program mean?
   - Are there syntax errors?



3. Code Generation (& Optimization)
   - Translate the program to an executable form.

mypgm.exe
101110100
001010001
111010000

Syntax Definition

- Backus-Naur form (BNF)

```
<assign>     := <operand> = <expression>
<expression> := <expression> <operator> <expression> | <term>
<term>       := <variable> | <literal>
<operator>   := + | - | * | / | ^ | %
```

- Standard for language definition since Algol (~1960)

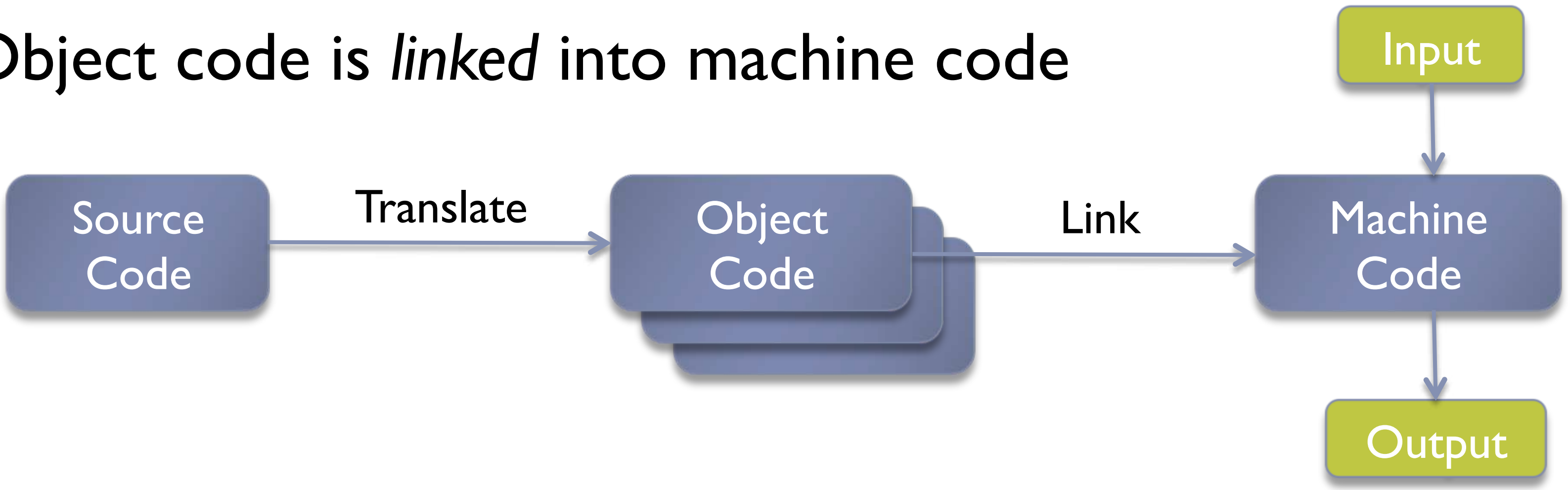
- Supports automation of lexical analysis, syntax analysis

Implementation Forms

- Compiled
  - source code => object code => machine code
- Interpreted
  - source code is processed directly by an interpreter
- Hybrid
  - 1) source code => intermediate code ("byte code")
  - 2a) intermediate code input by Virtual Machine (interpreter)
  - 2b) intermediate code compiled "just in time" to machine code

Compilation

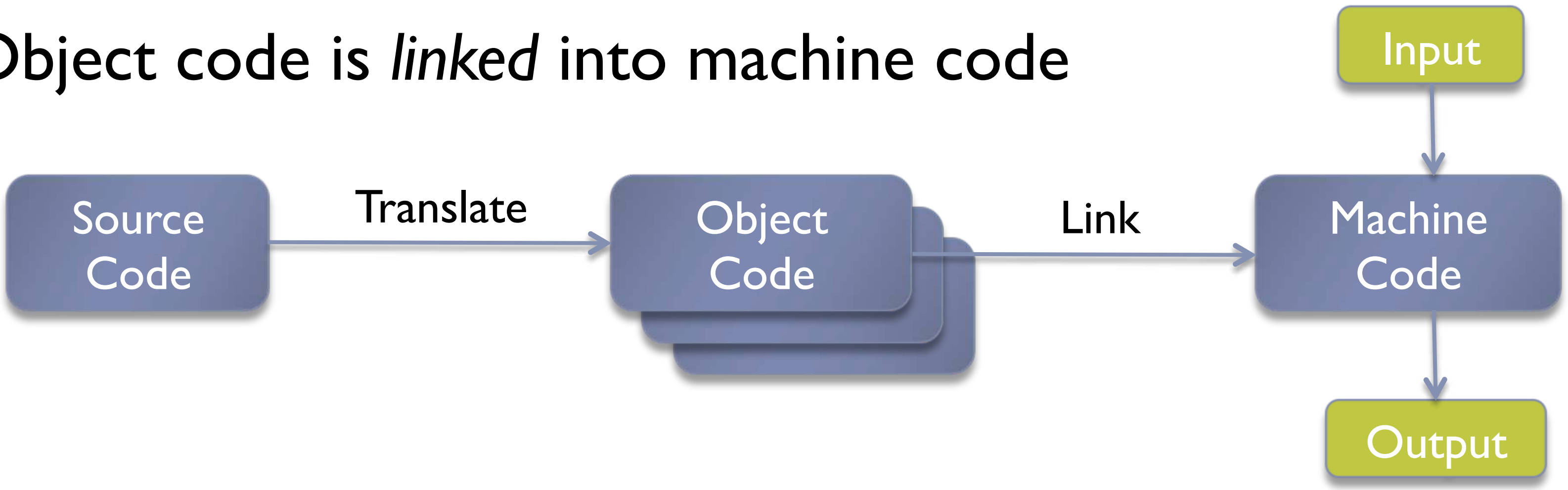
- Source code is *translated* to object code
- Object code is *linked* into machine code



- Examples: C, C++, Fortran
- Advantages?

Compilation

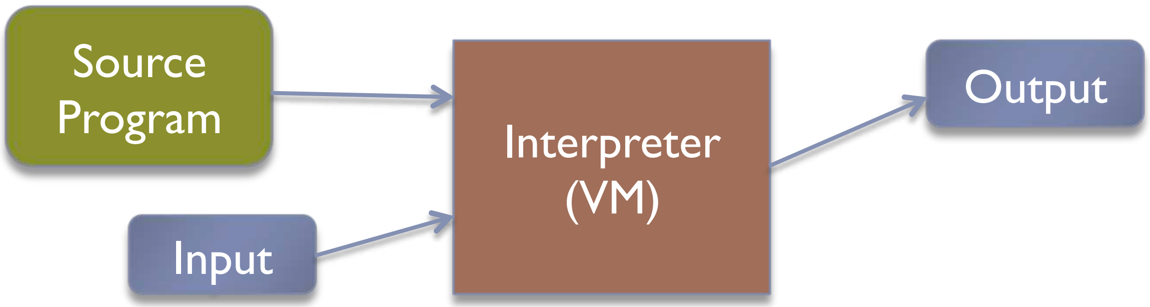
- Source code is *translated* to object code
- Object code is *linked* into machine code



- Examples: C, C++, Fortran
- Advantages:
  Highly efficient (fast) execution

Interpretation

- An interpreter (program) processes the source program

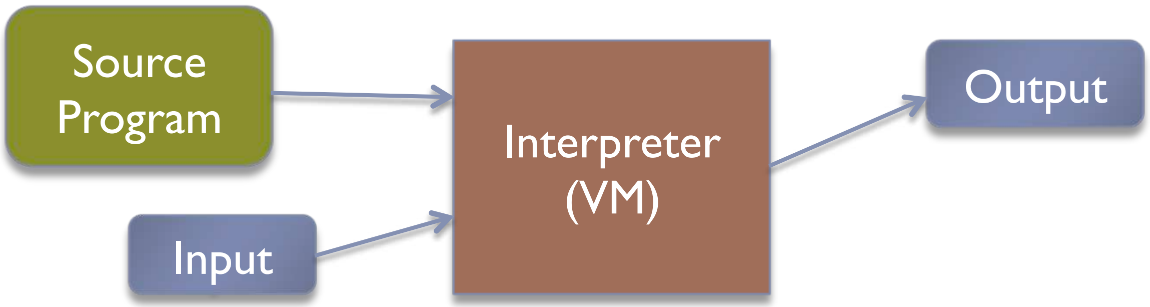


- The interpreter acts as a "virtual machine"

- Examples: JavaScript, Perl
- Advantages?

Interpretation

- An interpreter (program) processes the source program

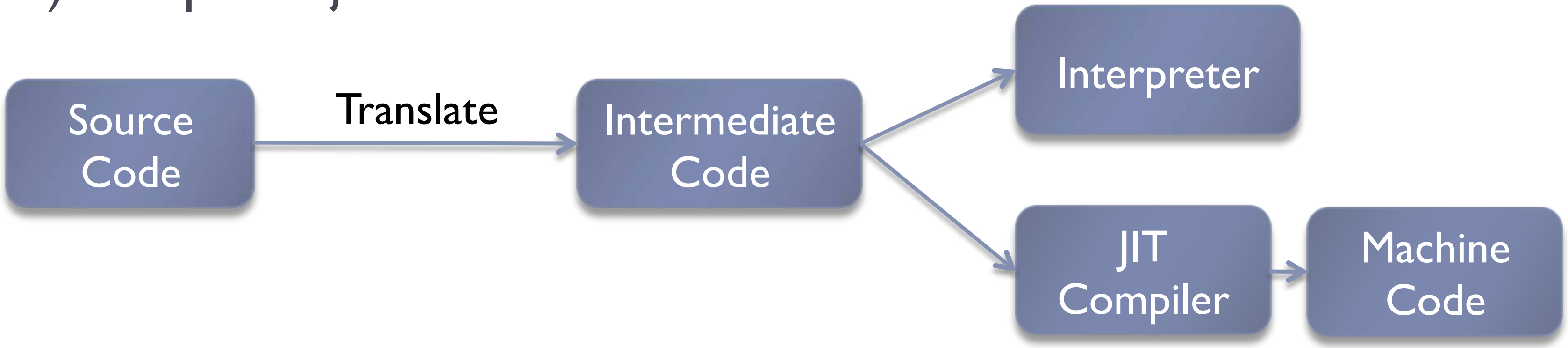


- The interpreter acts as a "virtual machine"

- Examples: JavaScript, Perl
- Advantages:
  Immediate execution - no wait for compilation
  Robust exception handling

Hybrid Implementation

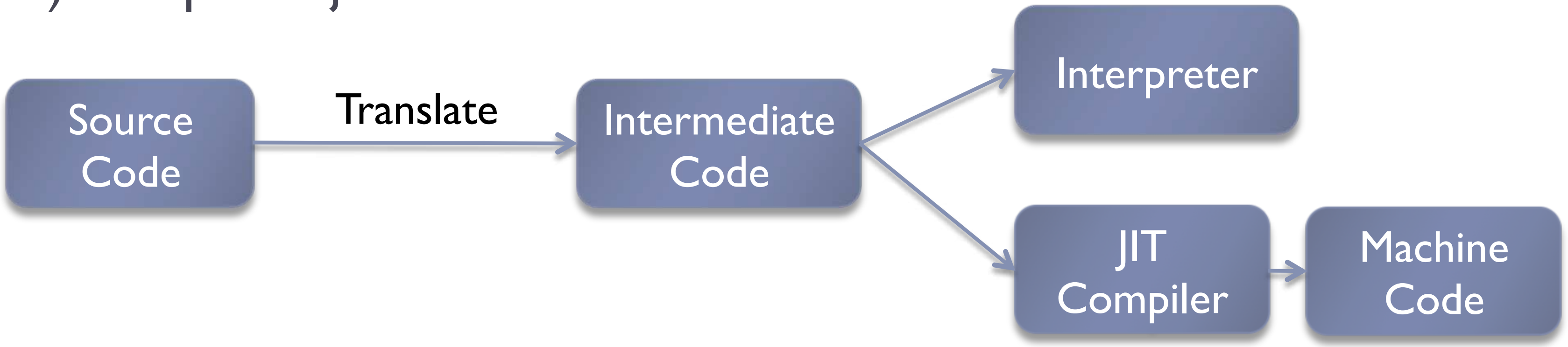
- Source code is first translated to intermediate code
- Intermediate code is
  - a) interpreted by a VM, or
  - b) compiled "just in time"



- Examples: Java (VM), C# .Net (JIT)
- Advantages?

Hybrid Implementation

- Source code is first translated to intermediate code
- Intermediate code is
  - a) interpreted by a VM, or
  - b) compiled "just in time"



- Examples: Java (VM), C# .Net (JIT)
- Advantages:
  - Most of the efficiency of pre-compiled code
  - Portability

Summary

- Language Generations
  - Machine Code, Assembly Language, High Level Language

- Program Compilation Phases:
  - Lexical Analysis => Parsing => Code Generation
  - Backus-Naur Form (BNF)

- Implementation Forms
  - Compilation (advantage: efficiency)
  - Interpretation (advantage: immediacy)
  - Hybrid (advantage: portability)
    - Virtual Machine
    - JIT Compilation