DATA 520
Introduction to Programming

Information History

Computer History

Computer Languages

Computer Science

- how problems are solved using a computer

  Man, cabbage, goat, wolf problem
  Day of the week for a date
  Travelling salesman: shortest route between 10 cities
  Play Chess

- logic very important

- algorithms vs. Brute Force solutions

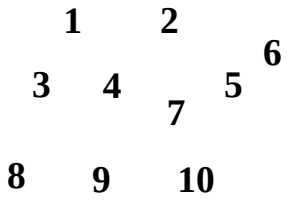
- can get VERY obscure

Algorithms

Brute Force vs. Algorithm solutions

Man, cabbage, goat, wolf problem - can do
http://jeux.lulu.pagesperso-orange.fr/html/anglais/loupChe/loupChe1.htm#

Day of the week for a date **- easy (data) way to solve?**



Travelling salesman challenge

10 cities = 10! possible routes = 3,628,800

20 cities = 20! possible routes = (2,432,902,008,176,640,000)

- computation time too long (77 years)

Play Chess

10^{120} possible games > 3 x 10^{90} grains of sand to fill universe

Algorithms

input: day, month, year

To determine the day of the week for a given **month**, **day**, and **year**:

1. Let **century_digits** be equal to the first two digits of the year.
2. Let **year_digits** be equal to the last two digits of the year.
3. Let **value** be equal to **year_digits** + floor(**year_digits** / 4)
4. If **century_digits** equals 18, then add 2 to **value**, else
   if **century_digits** equals 20, then add 6 to **value**.
5. If the **month** is equal to January and **year** is not a leap year, then add 1 to **value**, else,

   if the **month** is equal to February and the **year** is a leap year, then add 3 to **value**; if not a leap year, then add 4 to **value,** else,

   if the **month** is equal to March or November, then add 4 to **value**, else,

   if the **month** is equal to May, then add 2 to **value**, else,

   if the **month** is equal to June, then add 5 to **value**, else,

   if the **month** is equal to August, then add 3 to **value**, else,

   if the **month** is equal to October, then add 1 to **value**, else,

   if the **month** is equal to September or December, then add 6 to **value**,
6. Set **value** equal to (**value** + **day**) mod 7.
7. If **value** is equal to 1, then the day of the week is Sunday; else

   if **value** is equal to 2, day of the week is Monday; else

   if **value** is equal to 3, day of the week is Tuesday; else

   if **value** is equal to 4, day of the week is Wednesday; else

   if **value** is equal to 5, day of the week is Thursday; else

   if **value** is equal to 6, day of the week is Friday; else

   if **value** is equal to 0, day of the week is Saturday

FIGURE 1-8 Day of the Week Algorithm

value = numeric answer
floor function = truncate

- is it a leap year?

(if April
or July :
do nothing)

mod function = modulo
= remainder after division

Two choices:

```
if ...
 then ...
 else ...
```

More choices:

```
if ...
 then ...
 else if ...
```

Algorithms

Oops - we need an algorithm for leap years
Wikipedia:
The following pseudocode determines whether a year is a *leap year* or a *common year* in the Gregorian calendar (and in the proleptic Gregorian calendar before 1582). The *year* variable being tested is the integer representing the number of the year in the Gregorian calendar, and the tests are arranged to dispatch the most common cases first. Care should be taken in translating mathematical integer divisibility into specific programming languages.

if (*year* is **not** [evenly] divisible by 4) **then** (it is a common year)
else if (*year* is **not** divisible by 100) **then** (it is a leap year)
else if (*year* is not divisible by 400) **then** (it is a common year)
else (it is a leap year)

2016?
1900?

Computers use binary digits (bits)

Why, when we are used to base 10 numbers?

- electronic switches are either on or off

Any number in base 10 can be converted to base 2 (binary digits)

Base 10: digits 0…9, based on powers of 10:

$153_{10} = \mathbf{1}(10^2) + \mathbf{5}(10^1) + \mathbf{3}(10^0)$ (digits: 0 - 9)

$103_{10} = \mathbf{1}(10^2) + \mathbf{0}(10^1) + \mathbf{3}(10^0)$

Base 2: 0 or 1, based on powers of 2:

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1

$153_2 = \mathbf{1}(2^7) + \mathbf{0}(2^6) + \mathbf{0}(2^5) + \mathbf{1}(2^4) + \mathbf{1}(2^3) + \mathbf{0}(2^2) + \mathbf{0}(2^1) + \mathbf{1}(2^0)$

128 + 16 + 8 + 1

= 10011001

Computers use binary digits (bits)

Eight bits make a byte

10111001 = 153

00000001 = 1

Maximum value of a byte: 11111111 = 255

256 possible values, 0 … 255.
$256 = 2^8$

2^{16} = 65,536 = 64 kB

2^{32} = 4,294,967,296 = 4 GB

Numbers > 255 need more than one byte of storage.

Computers use binary digits (bits)

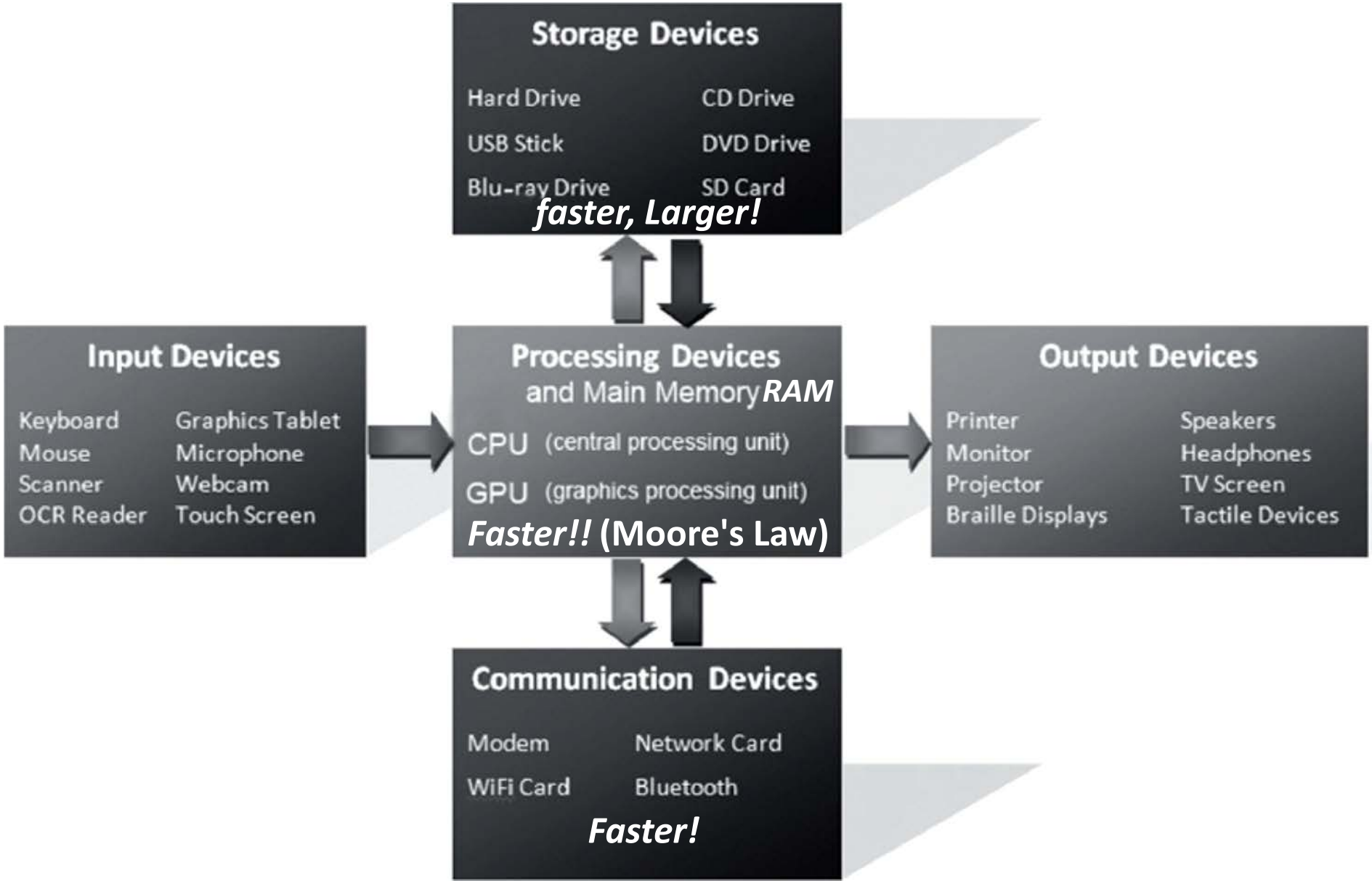
- for letters and keys on the keyboard too
ASCII key codes

A...Z = 97…122
a…z = 65…90

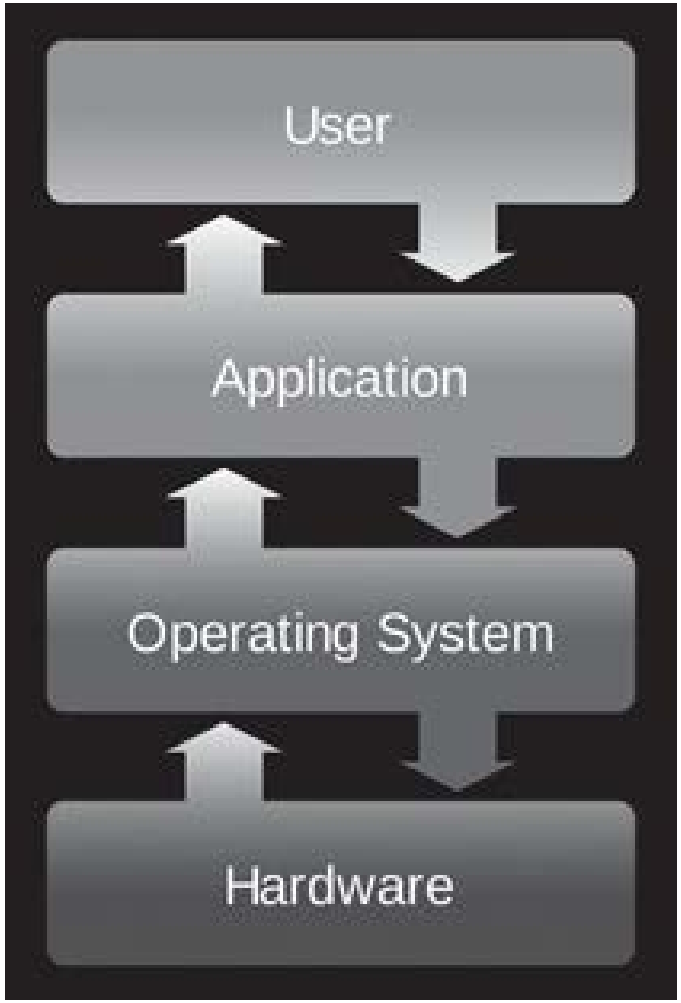
1 = 49
1 on number pad = 96

ESC = 27
backspace = 14
shift = 16
left arrow = 37

Computer hardware



Computer software



Chrome, Word, Excel, Python

Windows, OSX, iOS, Linux, Android

Processor, Motherboard, Storage

Software

A set of instructions that get executed on a computer

Computer programs involve syntax and semantics

syntax: spelling, word order, and punctuation

"Hello, are you how?"

semantics: meaning of words

"Colorless green ideas sleep furiously."

Programs

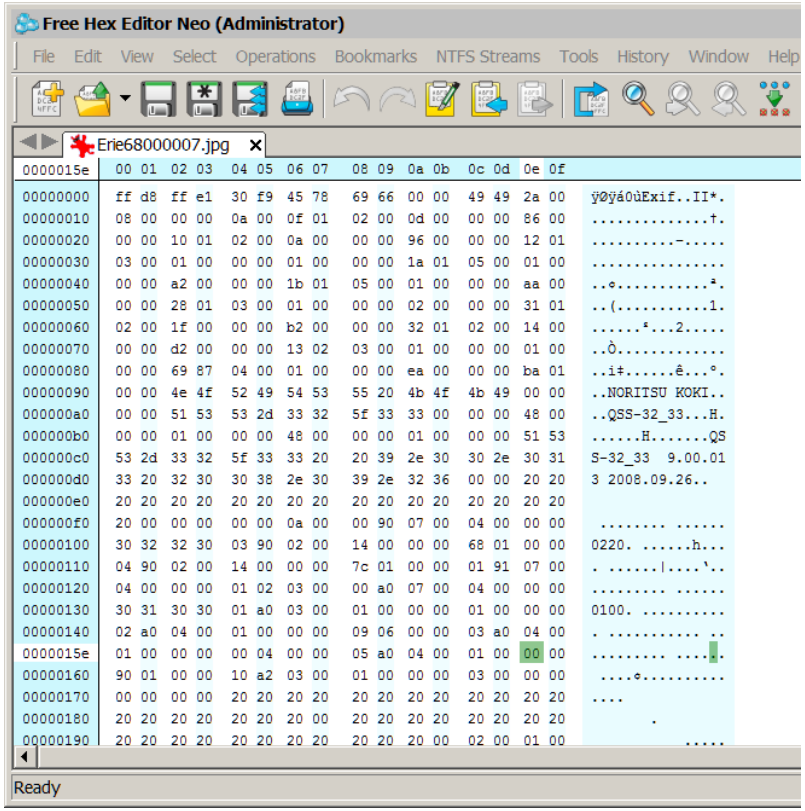
Programming languages: overwhelmingly English based

print, if… then… else…, etc.

- get translated into machine code (0101010101010)

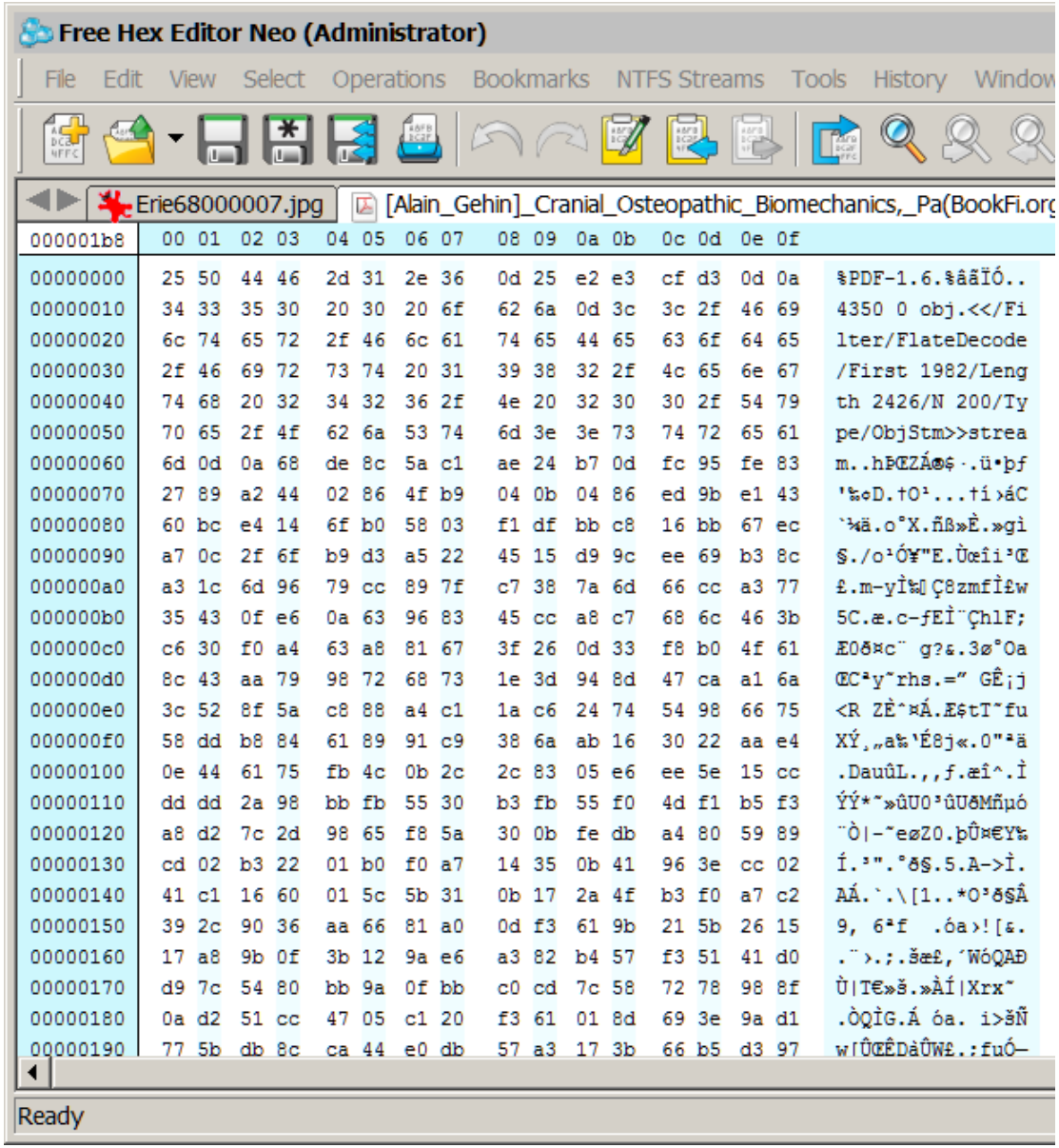
or hexadecimal

Base 16:
01… 09
and
0a…ff

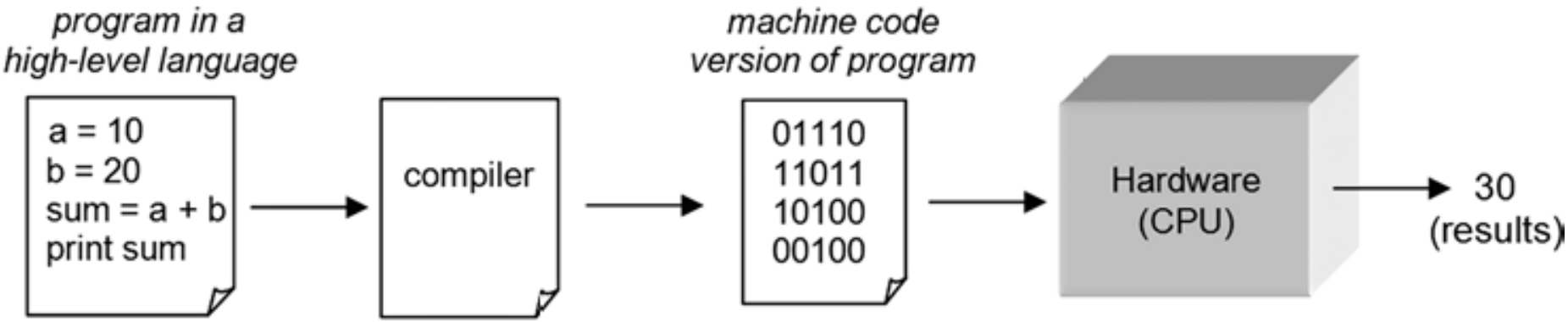


Programs

Hexadecimal



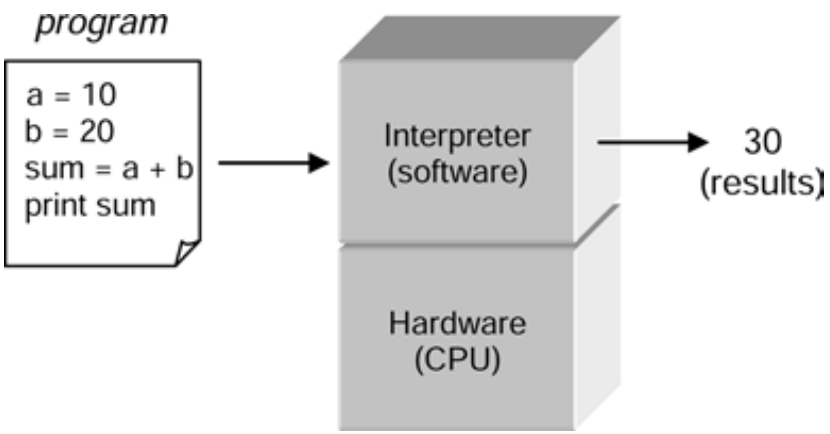
Programs



Some programming languages use a compiler

- generates machine-specific code
- fast
- limited platforms

Programs



Other programming languages use an interpreter

- use the interpreter for the language and computer
- interpreter executes the program
- easier for interactive programming (one line at a time)
- slower
- many platforms

Bugs

Why are they called "bugs"?

syntax error:

```
prnt('Hello, Erie')
```

Semantic errors aren't trapped by the program

```
#get the mean of three numbers
(num1 + num2 + num3) / 2.0

# print massage  <- also an error
print('Hellp, Eerie')
```

- the computer does not know what you want!
- it only does what you tell it to do
- debugging is always necessary

Computer Languages

1950's	Assembly language
1960's	Fortran
1970	Pascal
1972	C
1975	Altair Basic (Gates)
1983	GW-BASIC (MS)
1983	C++
1985	QuickBASIC (MS)
1986	Object Pascal
1987	Turbo Pascal
1987	Perl
1989	PowerBASIC
1989	Bash (Unix/Linux)
1991	Python
1991	Visual Basic
1992	Borland Pascal
1995	Java
1995	Javascript
2000	C#
2004	FreeBASIC

Computer Languages

Assembly language

```
00000030 B9FFFFFFFF
00000035 41
00000036 803C0800
0000003A 75F9
0000003C C3
```

```
;-----------------------------------------------------------
; zstr_count:
; Counts a zero-terminated ASCII string to determine its size
; in:   eax = start address of the zero terminated string
; out:  ecx = count = the length of the string

zstr_count:                     ; Entry point
    mov  ecx, -1                ; Init the loop counter, pre-decrement
                                ;  to compensate for the increment
.loop:
    inc  ecx                    ; Add 1 to the loop counter
    cmp  byte [eax + ecx], 0    ; Compare the value at the string's
                                ;  [starting memory address Plus the
                                ;  loop offset], to zero
    jne  .loop                  ; If the memory value is not zero,
                                ;  then jump to the label called '.loop',
                                ;  otherwise continue to the next line
.done:
                                ; We don't do a final increment,
                                ;  because even though the count is base 1,
                                ;  we do not include the zero terminator in the
                                ;  string's length
    ret                         ; Return to the calling program
```

Other Computer Languages

COBOL:

- earlier: hybrid assembly language

NOT `x = y` BUT: `MOVE x to y`

Still used;

New versions object-oriented

COBOL Code

```
DETERMINE-PMHP.                                                    04380074
043900    IF PMHP-FAC NOT = CC-FAC                                 04390074
044000       MOVE 'Y' TO EOF-PMHP                                  04400074
044100       GO TO EXIT-PMHP.                                      04410074
044200    INITIALIZE SORT-RECORD.                                  04420074
044300    MOVE PMHP-FAC  TO SORT-FAC.                              04430074
044400    MOVE PMHP-CASE TO SORT-CASE.                             04440074
044500    IF PMHP-INELIGIBLE-CODE NOT = ZERO                       04450074
044600       MOVE 5 TO SORT-PMHP-STATUS                            04460074
044700    ELSE IF PMHP-ACCEPT-DECLINE-FLAG = 2                     04470074
044800       MOVE 4 TO SORT-PMHP-STATUS                            04480074
044900    ELSE IF (PMHP-ENROLL-DSS-RESPONSE = 01 OR 02)            04490074
045000     AND PMHP-DISENROLL-DSS-DATE = ZEROES                    04500074
045100       MOVE 1 TO SORT-PMHP-STATUS                            04510074
045200    ELSE IF PMHP-DISENROLL-DSS-DATE NOT = ZERO               04520074
045300       MOVE 3 TO SORT-PMHP-STATUS                            04530074
045400    ELSE IF PMHP-ENROLL-EXTRACT-DATE = ZEROES                04540074
045500     AND PMHP-DISENROLL-REASON NOT = ZERO                    04550074
045600       MOVE 3 TO SORT-PMHP-STATUS                            04560074
045700    ELSE                                                     04570074
045800       MOVE 2 TO SORT-PMHP-STATUS                            04580074
045900       IF PMHP-CORRECTION-DATE > PMHP-ENROLL-EXTRACT-DATE    04590074
046000          MOVE 1 TO SORT-READY-RESEND
```

Other Computer Languages

COBOL:

```
//COBUCLG  JOB (001),'COBOL BASE TEST',                                 00010000
//             CLASS=A,MSGCLASS=A,MSGLEVEL=(1,1)                        00020000
//BASETEST EXEC COBUCLG                                                 00030000
//COB.SYSIN DD *                                                        00040000
 00000* VALIDATION OF BASE COBOL INSTALL                               00050000
 01000 IDENTIFICATION DIVISION.                                         00060000
 01100 PROGRAM-ID. 'HELLO'.                                             00070000
 02000 ENVIRONMENT DIVISION.                                            00080000
 02100 CONFIGURATION SECTION.                                           00090000
 02110 SOURCE-COMPUTER.  GNULINUX.                                      00100000
 02120 OBJECT-COMPUTER.  HERCULES.                                      00110000
 02200 SPECIAL-NAMES.                                                   00120000
 02210     CONSOLE IS CONSL.                                            00130000
 03000 DATA DIVISION.                                                   00140000
 04000 PROCEDURE DIVISION.                                              00150000
 04100 00-MAIN.                                                         00160000
 04110     DISPLAY 'HELLO, WORLD' UPON CONSL.                           00170000
 04900     STOP RUN.                                                    00180000
//LKED.SYSLIB DD DSNAME=SYS1.COBLIB,DISP=SHR                            00190000
//            DD DSNAME=SYS1.LINKLIB,DISP=SHR                           00200000
//GO.SYSPRINT DD SYSOUT=A                                               00210000
//                                                                      00220000
```

After submitting the JCL, the MVS console displayed:

```
19.52.48 JOB    3  $HASP100 COBUCLG  ON READER1     COBOL BASE TEST
19.52.48 JOB    3  IEF677I WARNING MESSAGE(S) FOR JOB COBUCLG  ISSUED
19.52.48 JOB    3  $HASP373 COBUCLG  STARTED - INIT  1 - CLASS A - SYS BSP1
19.52.48 JOB    3  IEC130I SYSPUNCH DD STATEMENT MISSING
19.52.48 JOB    3  IEC130I SYSLIB   DD STATEMENT MISSING
19.52.48 JOB    3  IEC130I SYSPUNCH DD STATEMENT MISSING
19.52.48 JOB    3  IEFACTRT - Stepname  Procstep  Program   Retcode
19.52.48 JOB    3  COBUCLG    BASETEST  COB       IKFCBL00  RC= 0000
19.52.48 JOB    3  COBUCLG    BASETEST  LKED      IEWL      RC= 0000
19.52.48 JOB    3  +HELLO, WORLD
19.52.48 JOB    3  COBUCLG    BASETEST  GO        PGM=*.DD  RC= 0000
19.52.48 JOB    3  $HASP395 COBUCLG  ENDED
```

Other Computer Languages

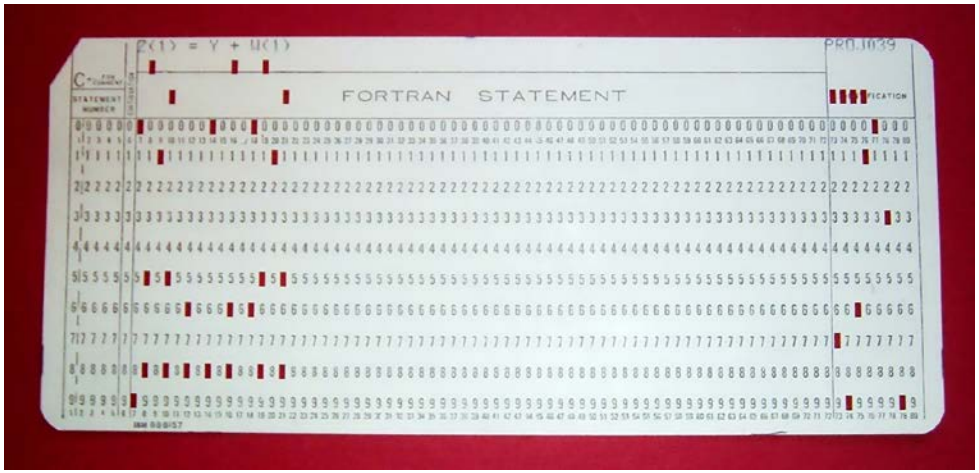
FORTRAN:

- ran almost as fast as assembly language
- needed compiler
- more efficient code (1/20 of the typing)

Fortran 77, Fortran 95

- ANSI standards

```
      K=1
6     IF (K.EQ.11) GO TO 8
      READ,I,J
      IF (J.GT.I) GO TO 65
      GO TO 66
65    WRITE(6,6002)J,I
6002  FORMAT(' ',I3,' IS GREATER THAN ',I3)
      K=K+1
      GO TO 6
66    WRITE(6,6001)I,J
6001  FORMAT(' ',I3,' IS GREATER THAN ',I3)
      K=K+1
      GO TO 6
8     CALL EXIT
      END
```



Other Computer Languages

BASIC

```
SUB MatrixInversion(A() AS DOUBLE, M AS LONG, Determinant AS DOUBLE)
    ' Gauss reduction inversion method.
    ' M is the order of the square matrix A()
    ' A() inverse is returned in A().
    ' Determinant is returned.
    LOCAL I, J, K, L AS LONG, T AS DOUBLE, Pivot AS DOUBLE
    Determinant = 1
    FOR J = 1 TO M
        Pivot = A(J,J) : A(J,J) = 1
        Determinant = Determinant * Pivot
        IF Determinant = 0 THEN MSGBOX "Matrix singular " _
          +  "- cannot invert",,"Problem": EXIT SUB
        '
        ' Divide pivot row with pivot element.
        FOR K = 1 TO M : A(J,K) = A(J,K) / Pivot : NEXT
        FOR K = 1 TO M
            ' Reduce the non pivot rows.
            IF K <> J THEN
                T = A(K,J) : A(K,J) = 0
                FOR L = 1 TO M : A(K,L) = A(K,L) - A(J,L) * T : NEXT
            END IF
        NEXT
    NEXT
END SUB
```

Other Computer Languages

C, ugly?

```
        CachePara(psel->doc, psel->cpFirst);
        if (psel->cpLim >= caPara.cpLim)
            {
            ca = psel->ca;
    /* expand selection to integral paragraphs */
            FUpdateHplcpad(ca.doc);
            fForwardSave = psel->fForward;
            ExpandOutlineCa(&ca, fFalse);
            if (FInTableDocCp(ca.doc, cpLimM1 = ca.cpLim - 1) &&
                    FInTableDocCp(ca.doc, ca.cpFirst))
                {
                CacheTc(wwNil, ca.doc, ca.cpFirst, fFalse, fFalse);
                caTable = caTap;
                cpFirstTable = caTap.cpFirst;
                if (!FInCa(ca.doc, cpLimM1, &vtcc.ca) &&
                        (FInCa(ca.doc, cpLimM1, &caTable) || FParasUpToCpInTable(&caTable, cpLimM1)))
                    {
                    CpFirstTap(ca.doc, cpLimM1);
                    cpLimTable = caTap.cpLim;
                    SelectRow(psel, cpFirstTable, cpLimTable);
                    return;
                    }
                }
            Select(psel, ca.cpFirst, ca.cpLim);
            psel->cpAnchor = fForwardSave ? ca.cpFirst : ca.cpLim;
            psel->fForward = fForwardSave;
            psel->sty = styPara;
            SetSelCellBits(psel);
            }
        }
```

Other Computer Languages

C++, ugly?

```
BOOL CMymfc29BAuto::DisplayDialog()
{
    // TODO: Add your dispatch handler code here
    TRACE("Entering CMymfc29BAuto::DisplayDialog %p\n", this);
    BOOL bRet = TRUE;
    AfxLockTempMaps();  // See MFC Tech Note #3
    CWnd* pTopWnd = CWnd::FromHandle(::GetTopWindow(NULL));
    try
    {
        CPromptDlg dlg /*(pTopWnd)*/;
        if (m_vaTextData.vt == VT_BSTR)
        {
            // converts double-byte character to single-byte character
            dlg.m_strData = m_vaTextData.bstrVal;
        }
        dlg.m_lData = m_lData;
        if (dlg.DoModal() == IDOK)
        {
            m_vaTextData = COleVariant(dlg.m_strData).Detach();
            m_lData = dlg.m_lData;
            bRet = TRUE;
        }
        else
        {
            bRet =  FALSE;
        }
    }
    catch (CException* pe)
    {
        TRACE("Exception: failure to display dialog\n");
        bRet =  FALSE;
        pe->Delete();
    }
    AfxUnlockTempMaps();
    return bRet;
}
```

Other Computer Languages

Pascal

```
File   Edit   Run   Compile   Options   Debug   Break/watch
                          Edit
     Line 15   Col 39  Insert Indent        Unindent * D:NONAME.PAS
program KenLovesTurboPascal;
uses
   crt;
var
   age: Integer;
   name: String;
   message: String;
begin
   ClrScr;
   name := 'Ken Egozi';
   age := 30;
   if age < 10 then
       message := ' loves Turbo Pascal'
   else
       message := ' loved Turbo Pascal';
   write (name);
   writeln (message);
end.
                          Watch

F1-Help  F5-Zoom  F6-Switch  F7-Trace  F8-Step  F9-Make  F10-Menu
```

Python

Invented in 1991

- named after Monty Python's Flying Circus

- simple syntax
- VERY readable code

- has many standard modules and many extensions

A Python Code Sample

```
x = 34 - 23                # A comment.
y = "Hello"                # Another one.
z = 3.45
if z == 3.45 or y == "Hello":
    x = x + 1
    y = y + " World"       # String concat.
print x
print y
```

Installing Python

https://www.python.org/downloads/

Find your OS

(https://www.python.org/downloads/mac-osx/)

We will use Python 3.5.2 (lab) or 3.6.2 (download?)

- includes IDLE, an IDE (Integrated Development Environment)

Next time

Read chapter 2 in Gries book