FORTRAN

The Only Real Computer Programming Language

FORmula TRANslation

- First compiler written from scratch 1954-1957 by an IBM team lead by John W. Backus. Just called FORTRAN
- FORTRAN II (1958)
- FORTRAN III (1958) never released.
- FORTRAN IV (1961)
- FORTRAN 66 (first standard high level language.)

FORmula TRANslation

- FORTRAN 77 – still used! (In fact, the compiler on esus is f77.)
  - DO loops with decreasing loop control
  - Block if statements IF…THEN…ELSE…ENDIF
  - Pre-test of DO loops
  - CHARACTER data type
  - Apostrophe delimited character string constants.
  - Main program termination without a STOP statement.

FORmula TRANslation

- FORTRAN 88 – wasn't released until 1990 so name was changed to FORTRAN 90.
  - Free format source code (column independent)
  - Modern control structures (case & do-while)
  - Records/structures (derived data types)
  - Powerful array notation
  - Dynamic memory allocation
  - Operator overloading
  - Keyword argument passing
  - INTENT (in, out, inout)
  - Modulels

FORmula TRANslation

- FORTRAN 95
  - FORALL
  - Partial nesting of FORALL and WHERE
  - Masked ELSEWHERE
  - Pure procedures
  - Elemental procdures
  - Revised MINLOC and MAXLOC
  - Extensions to CEILING and FLOOR
  - Various other improvements.

Fortran 2003 and Fortran 2008!

- See following link for Fortran 2003
  - Fortran 2003
- See following link for Fortran 2008
  - Fortran 2008

FORmula TRANslation

- My first FORTRAN book was plain original FORTRAN.
- Shortly, it became FORTRAN II and I began programming on an IBM 1620 card based machine. (circa 1963 at Whitman College in Walla Walla, WA)
- First actual program was a correlation coefficient calculation.

Source Format

- Fixed Source Form
  - The most compatible with older systems.
  - Col 1 (C or *) means comment
  - Col 2 – 5 statement labels
  - Col 6 Continuation column
  - Col 7-72 Statements
  - Col 73-80 Sequence Numbers

Fixed Source Example

```
C EXAMPLE FIXED SOURCE CODE
C
      PROGRAM TEST1
      INTEGER X,Y,Z
      DATA X,Y,Z /0,2,4/
      DO 100 X = 1,10
      PRINT *, X
  100     CONTINUE
      CALL EXIT
      END
```

Continuation

- Since you can't put characters in 72-80, you need to extend statements by putting a character in column 6.
- print *, "The name is ", NAME,
- 1 "The address is", ADDRESS,
- 2 "The city is ", CITY
- You must line these continuations up carefully in the fixed source form.

Sequence Numbers??

- Used with cards. (The "researcher in the road" story)
- You could take unordered cards and run them through the sorter machine and put your code back "in order".
- Funny to think about today! ☺
- I think there still may be some card readers in ITC (really dusty!)

Source Format

- Free Source Form
  - No column restrictions
  - ! Begins a comment anywhere
  - Continuation indicated by an & at the end of a line.

Free Source Example

```
C EXAMPLE FIXED SOURCE CODE
C
PROGRAM TEST1
INTEGER X,Y,Z
DATA X,Y,Z /0,2,4/
DO 100 X = 1,10
PRINT *, X
100  CONTINUE
CALL EXIT
END
```

Case Insensitive!!!

- Ray, RAY, ray are all the **SAME** variable name.
- Try to be consistent to improve readability.

FORTRAN 77

- I'll describe FORTRAN 77 since it is what I have used the most!
- There still is a LOT of FORTRAN code out there.
- "We've got this old FORTRAN program that we'd like you to debug, interested?"
  - >=FORTRAN 66?..... Yes!
  - <FORTRAN 66?..... No!

F77 Types

- TYPE_STATEMENT Variable Names
  - Some examples follow:
  - INTEGER FRED, SUZY, CLIPIT
    - assumes 4 byte integers.
  - INTEGER*2 NAME, A,B,C
    - *2 means 2 byte integer.
  - INTEGER*4 BIGGER, BETTER
    - *4 means 4 byte integer.
  - Must be before executable statements!

INTEGER ARRAYS

- INTEGER I(10), C(10,2)
- Arrays are unit indexed by default. I above would be I(1) through I(10).
- Two dimensional arrays are stored in column major order! C(1,1), C(1,2), C(2,1), C(2,2), …

REAL types

- REAL FRED, SAM, GEORGE
- REAL*4 LUCY, ANN, MARY
- REAL*8 BIGGER, BETTER, BEST

CHARACTER TYPES

- CHARACTER MID, NAME
- Single character only stored in NAME!
- If you want a string, you need an array:
  - CHARACTER FIRST(25)

LOGICAL TYPE

- LOGICAL list1
- LOGICAL*4 list2
- LOGICAL*1 list3
- LOGICAL constants
  - .TRUE.
  - .FALSE.

DOUBLE PRECISION TYPE

- DOUBLE PRECISION list
- Same as REAL*8

Three Basic Structures

- Sequence
- Selection
- Repetition
- Any single thread program can be written with just combinations of these three structures!

SEQUENCE

- One statement follows the other.
- Statements can have a numeric label.
- The infamous GOTO can cause a change in direction.
- MUCH ABUSED in my days!
- Leads to spaghetti code!

SELECTION

- Old fashioned arithmetic-IF
- IF(arith_expression) stn1,stn2,stn3
  - Arith_expression evaluated,
  - If results < 0, go to statement stn1
  - If results = 0, go to statement stn2
  - If results > 0, go to statement stn3
- Large logic set very hard to read

ARITHMETIC IF EXAMPLE

```
      IF (X – 10) 10, 10, 20
10    IF (Y – 20) 15, 30, 30
15    CONTINUE
20    PRINT *, X
30    CONTINUE
      PRINT *, Y
      …
```

ARITHMETIC IF EXAMPLE

```
      IF (X – 10) 10, 10, 20
10    IF (Y – 20) 15, 30, 30
15    CONTINUE
20    PRINT *, X
30    CONTINUE
      PRINT *, Y
      …
```

What values of X and Y cause this statement to be executed?

IF - THEN - ENDIF

```
IF (X .EQ. 10) THEN
 PRINT *, “X = 10”
 ENDIF

IF(Y .LT. 20) THEN
 PRINT *, “Y < 20”
 ENDIF
```

COMPARISON OPERATORS

- .LT. LESS THEN
- .LE. LESS THEN OR EQUAL
- .GT. GREATER THEN
- .GE. GREATER THEN OR EQUAL
- .EQ. EQUAL TO
- .NE. NOT EQUAL TO

LOGICAL OPERATORS

- .AND. AND
- .OR. OR
- .NOT. NOT
- IF(.NOT.(2.GT.3 .AND. 3.LT.4))
- IF(2.0 .LT. 3.0 .AND. 4.0 .GT. 3.0)

IF - THEN - ELSE - ENDIF

```
IF (A .LT. B  .AND.  C .GT.  D) THEN
  PRINT *, "THIS IS THE TRUE BRANCH"
  PRINT *, A,B,C,D
ELSE
   PRINT *, "THIS IS THE FALSE BRANCH"
   PRINT *, A,B,C,D
ENDIF
```

REPETITION

- This is one of the most misused non-structured components of FORTRAN.
- You can have a GOTO inside an if branch backwards and make a loop.
- You can just code a GOTO that branches back without an if (permanently branches)
- Or, you can use a more structured DO.

DO STATEMENT

- DO label var = start, end, increment
  - label is a numeric statement label
  - Start is an expression that evaluates to the starting value for the var.
  - End is an expression that evaluates to the ending value for the var.
  - Increment is what change is applied to the var each time through the loop.

EXAMPLE DO

```
      DO 120 K = 3, 10, 2
      PRINT *, "K= ", K
120   CONTINUE
```

What prints?

EXAMPLE DO

```
      DO 120 K = 3, 10, 2
      PRINT *, "K= ", K
120   CONTINUE
```

What prints?

K= 3

K= 5

K= 7

K= 9

ARITHMETIC OPERATORS

- * multiplication
- / division
- + addition, unary positive
- - subtraction, unary negative
- ** exponentiation

REAL ARITHMETIC

- Providing all variables and constants in the expression are real, real arithmetic will be carried out as expected, with no decimal places being truncated.

INTEGER ARITHMETIC

- Providing the expression has all integers, subtraction, addition, multiplication and exponentiation will prove no problem. However integer division is somewhat different than normal division with real values. Integer division ignores the fractional part. Any decimal places are truncated.

MIXED-MODE ARITHMETIC

- Mixed mode arithmetic is when an expression contains both reals and integers. If ANY of the operands are real then result of the operation will be real. However, mixed mode arithmetic should be used with extreme care. You may think you have a real operand when in reality you have two integer operands.

MIXED MODE EXAMPLES

- 5 / 2 * 3.0 =

MIXED MODE EXAMPLES

- 5 / 2 * 3.0 = 6.0
- 3.0 * 5 / 2 =

MIXED MODE EXAMPLES

- 5 / 2 * 3.0 = 6.0
- 3.0 * 5 / 2 = 7.5

ARITHMETIC FUNCTIONS

- ABS absolute value
- COS cosine
- DBL double
- DPROD dp product
- EXP exponentiation
- INT integer
- LOG logarithm
- MAX max of a list
- MIN min of a list
- MOD int division
- NINT round to int
- REAL real
- SIN sine
- SQRT square root

Input Output Statements

- READ
- WRITE
- PRINT
- FORMAT CONTROLS IN SEPARATE STATEMENT
- UNIT NUMBERS TO DIRECT IN/OUT
- UNFORMATTED * FOR EASE OF USE.

READ * / PRINT *

- ASSUMES A UNIT (KEY BOARD)
- ASSUMES A FORMAT (TYPE CONTROLLED)
- E.G.
  - INTEGER A,B,C
  - READ * A,B,C
  - PRINT * A,B,C

  1 13 15 ← input

  1 13 15 → output

READ * / PRINT * (cont.)

- INTEGER A,C
- REAL B
- READ * A,B,C
- PRINT * A,B,C
- 12 4.5 -18 ← input
- 12 4.5 -18 → output
- Pretty straight forward!

Formatted I/O

- Add a FORMAT statement
  - Usually in a separate statement
  - ```
        I = 128
        PRINT 9000, I
    9000 FORMAT(1X, 'I = ', I8)
    ```
  - `I =  128` ← output


    ```

Format Codes

- A character string
- B blank handling
- D double precision
- E single precision
- F single precision
- G generic (d,f,i,…
- H literal strings
- I integer
- L logical values
- P scaling factor
- S sign control
- T tabbing control
- X horizontal space
- Z hexadecimal
- / vertical skipping
- ' enclosed literal
- : termination

Some Examples

- 100 FORMAT(1X, 3I5)
- 200 FORMAT(1X, F6.2, E12.5)
- First character is form control
  - ' ' (blank) skip one line before printing
  - '1' (one) skip one page before printing
  - (the 8 inch listing story)
  - '0' (zero) double space

Spacing

- Depends on the type of conversion.
- E.g. A format
  - If the field width is shorter than the number of characters stored in the variable, characters are truncated on the right.
  - If the field width is longer than the number of characters stored in the variable, the field is padded on the left with blank characters.

Sub Programs

- SUBROUTINE SAMPLE(R,S,X,Y)
- INTEGER R,S
- REAL X,Y
- … body of subroutine
- RETURN
- END

Function

```
INTEGER FUNCTION TEST(T,U,V)
REAL T,U
LOGICAL V
... body of function
TEST = 1 + 2
RETURN
END
```

More On Arrays

- INTEGER ARY(1:10)
- INTEGER BARY(-5:5)
- REAL*4 ARY2D(1:4,1:5)
- REAL*8 MYARY(-2:2,-4:4)
- INTEGER CUBE(1:3,1:5,1:10)
- INTEGER MYCUBE(-3:2,-4:2,-3:10)

Passing Arrays As Parameters

- INTEGER X(10)
- CALL SUB1(X)

- SUBROUTINE SUB1(A)
- INTEGER A(10)
- A(1) IS SAME AS X(1)

Passing Arrays As Parameters

- INTEGER X(10)
- CALL SUB2(X)

- SUBROUTINE SUB2(B)
- INTEGER B(7)
- B(1) IS SAME AS X(1)
- CAN REFERENCE FIRST 7 ELEMENTS OF X.

Passing Arrays As Parameters

- INTEGER X(10)
- CALL SUB3(X(3))

- SUBROUTINE SUB3(C)
- INTEGER C(4)
- C(1) IS SAME AS X(3)
- CAN REFERENCE 4 ELEMENTS OF X.

Passing Arrays As Parameters

- INTEGER R(2,3)
- CALL SUB4(R)

- SUBROUTINE SUB4(T)
- INTEGER T(2,3)
- T(1,1) IS SAME AS R(1,1)
- CAN REFERENCE ALL 6 ELEMENTS OF R.

Passing Arrays As Parameters

- INTEGER R(2,3)
- CALL SUB5(R)

- SUBROUTINE SUB5(U)
- INTEGER U(3,2)
- U(1,1) IS SAME AS R(1,1)
- CAN REFERENCE ALL 6 ELEMENTS OF R.
- U(1,1)=R(1,1)
- U(2,1)=R(2,1)
- U(3,1)=R(1,2)
- U(1,2)=R(2,2)
- U(2,2)=R(1,3)
- U(3,1)=R(2,3)

Passing Arrays As Parameters

- INTEGER R(2,3)
- CALL SUB6(R(1,2))

- SUBROUTINE SUB6(V)
- INTEGER V(2)
- V(1) IS SAME AS R(2,1)
- V WILL REFERENCE THE 2ND COLUMN OF ARRAY R.
- V(1)=R(1,2)
- V(2)=R(2,2)

Array Name Used Without Subscript

- In an argument or parameter list
- In COMMON or type statement
- In an EQUIVALENCE statement
- In a DATA statement
- In the list of an input or output statement
- As the unit identifier for an internal file in an input or output statement
- As a format identifier in an input or output statement

Assigned GOTO

- ASSIGN stno TO name
- GOTO name, (stno1, stno2, … stnok)

Call Subroutine

- CALL name (argument list)

Common Statement

- COMMON /common name/ common list
- COMMON /SAM/ A,B,C
- Same statement in main program and in subroutines.
- Two names for the same thing.

COMPLEX

- Built in to FORTRAN
- COMPLEX*8 LIST
- COMPLEX*16 LIST

Implied Do Loop

- Used in I/O
- READ *, N, (PRIME(I), I=1,N)
- WRITE *, (PRIME(J),J=1,3)

ELSEIF

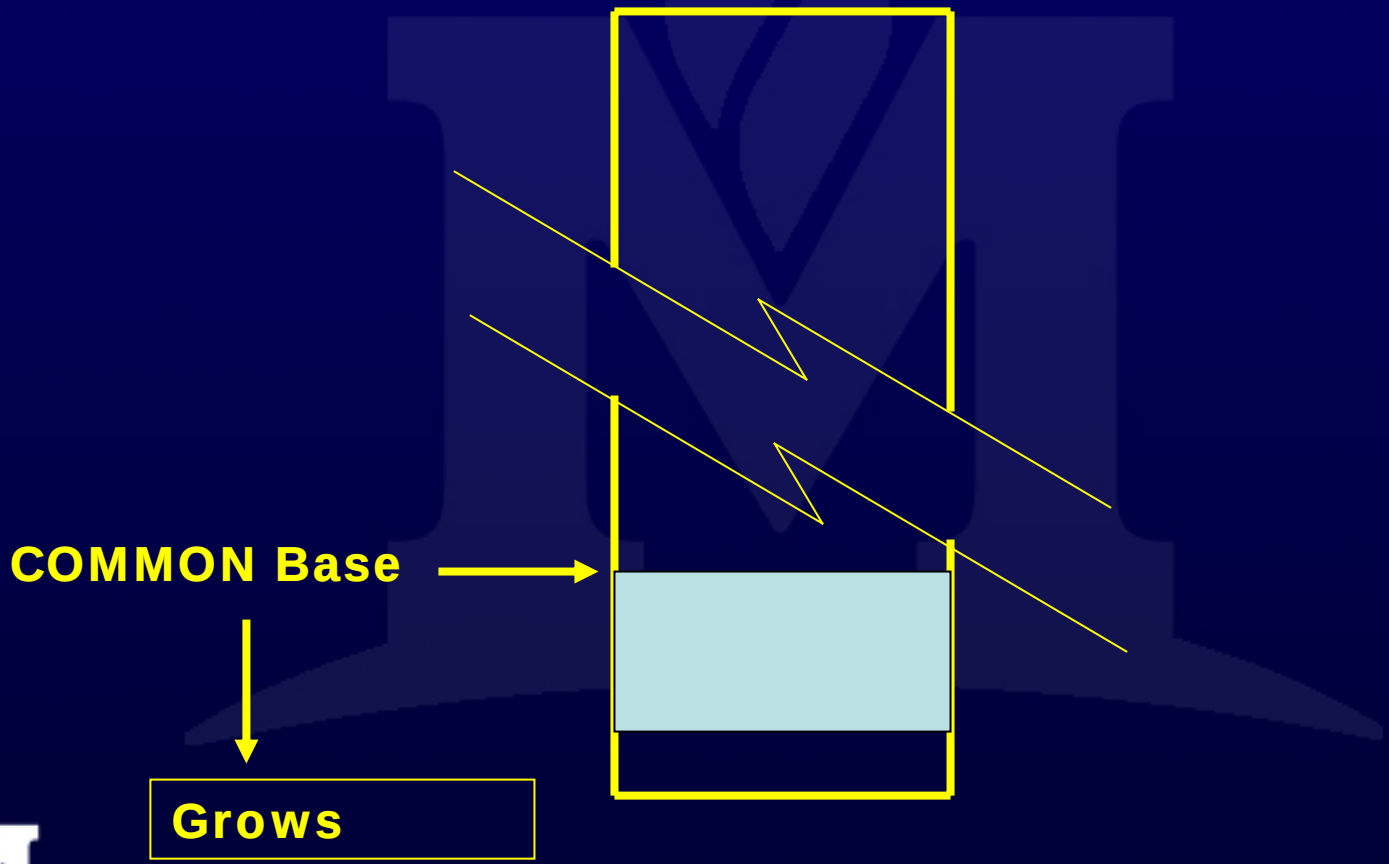
- IF (logical expression) THEN
- statements s1
- ELSEIF (logical expression) THEN
- statements s2
- ELSE
- statements s3
- ENDIF

Equivalence

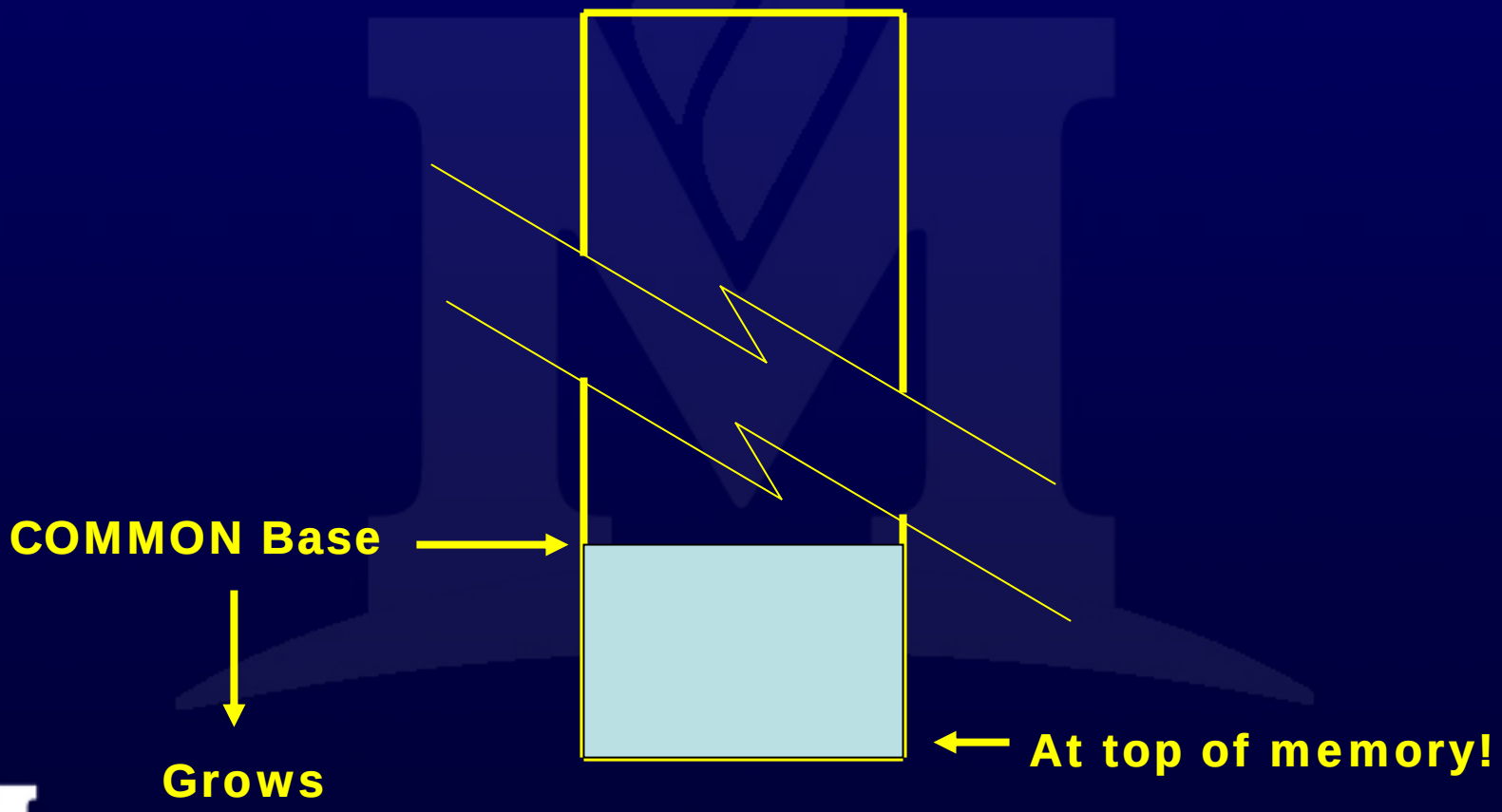
- EQUIVALENCE (A,B)
- Alias naming.
- INTEGER A(4),B(2),C(2)
- EQUIVALENCE (A(1),B(1)),(A(3),C(1))

A Last FORTRAN Story

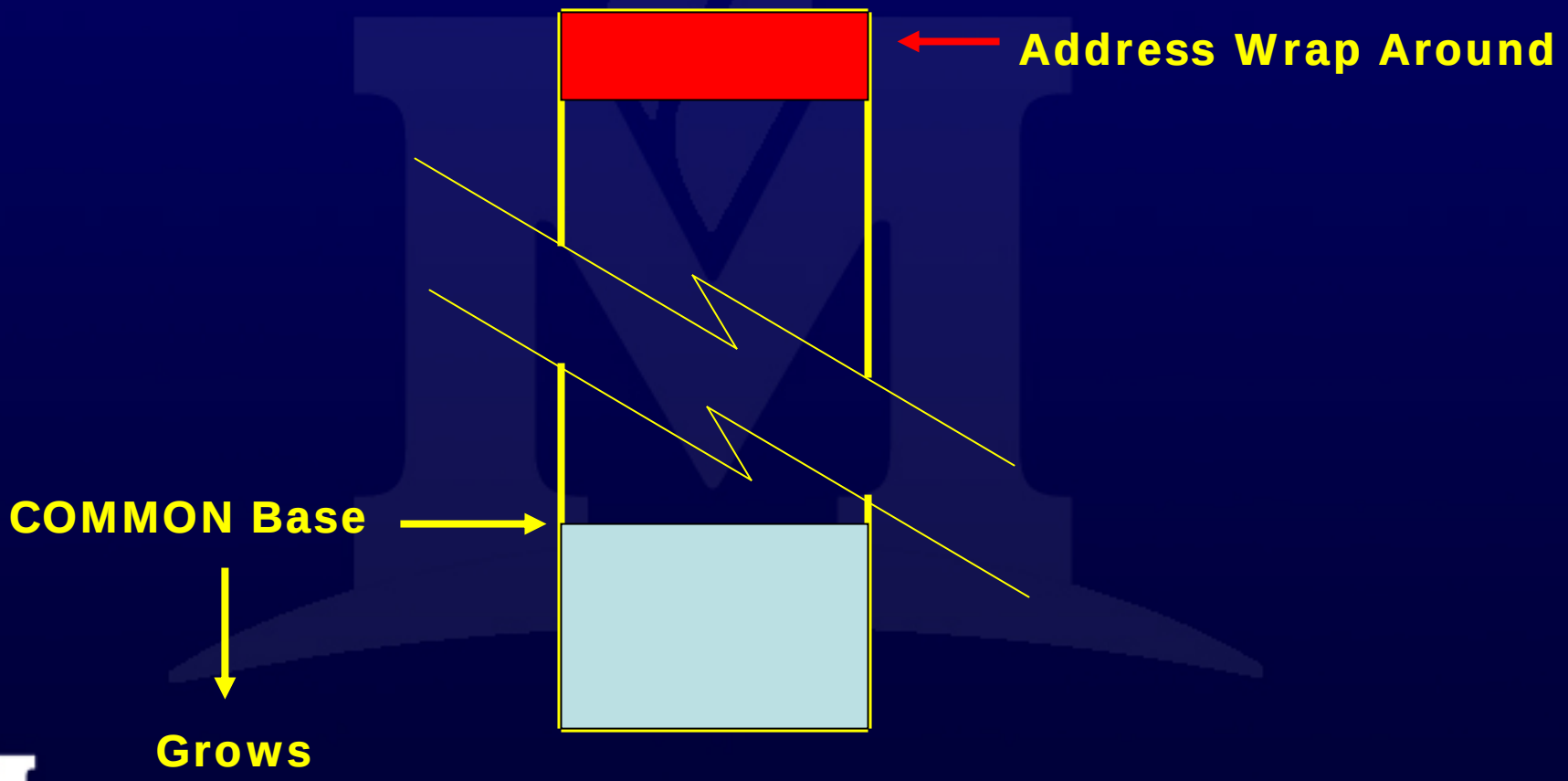
A Last FORTRAN Story



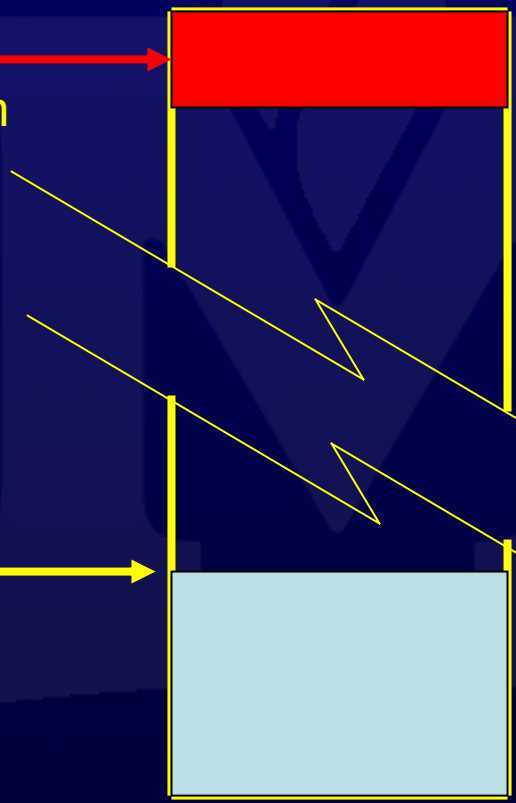
A Last FORTRAN Story



A Last FORTRAN Story



A Last FORTRAN Story



(00007) Program Counter (instruction address register)

HALT!

COMMON Base

Grows

FORTRAN 2000 ? Or F?

- http://www.fortran.com/F/
- http://www.fortran-2000.com