# Introdução ao Fortran 90

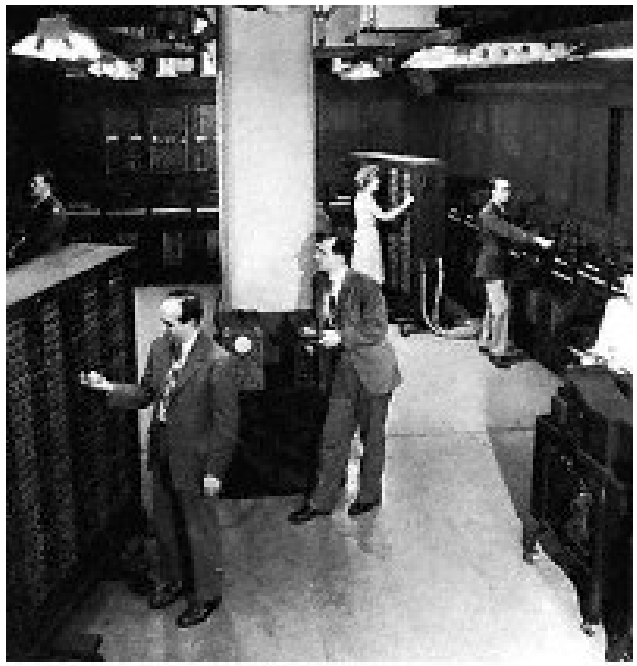
Alexandre Diehl

Departamento de Física – UFPel

## 1943-1953: Computador com Programa Fixo

ENIAC (Electronic Numerical Integrator and Computer)

- Início do Projeto: 1943
- Projeto Completo: 1946
- Programação: plug board e switches
- Velocidade: 5000 operações/segundo
- Input/Output: cartões, luzes, switches, plugs
- Área ocupada: 100 $m^2$



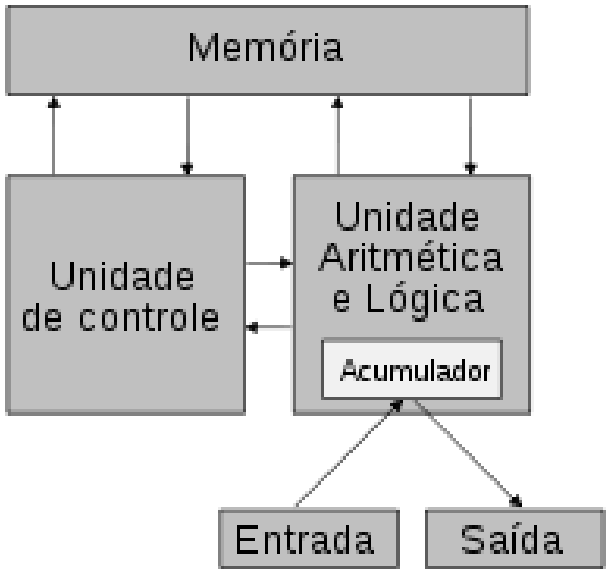
Programação em Linguagem de Máquina (0 e 1) → Baixo Nível

## 1946: Computador de Programa Armazenado

## Arquitetura de von Newmann:

Máquina digital capaz de armazenar programas no mesmo espaço de memória que os dados.

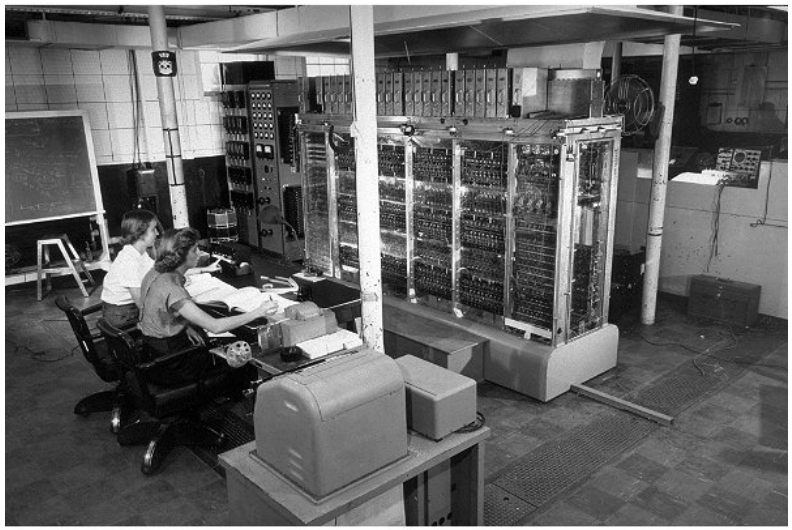
- Memória
- Unidade Central de Processamento (CPU)
  - Unidade Aritmética e Lógica (ULA)
  - Unidade de Controle (CU)
  - Conjunto de Registradores
- Dispositivos de Entrada e Saída



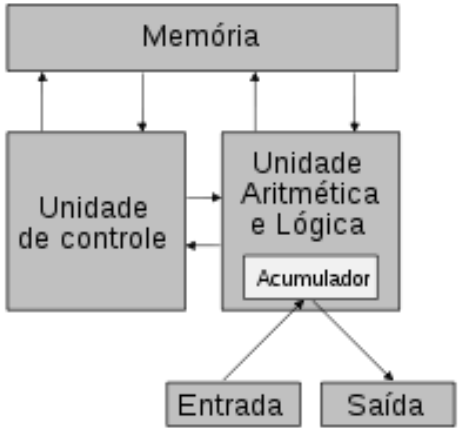
## 1952-1957: Computador de Programa Armazenado

MANIAC I (Mathematical Analyzer, Numerical Integrator, and Computer)

- Peso: 500 kg
- Capacidade de armazenar programas
- Usado no projeto Manhatan para a bomba de hidrogênio
- Usado para o cálculo de novas partículas sub-atômicas (1953-1954)
- Primeiro computador a jogar xadrez



## Estrutura da CPU




- ULA

  Coleção de circuitos que realizam as operações (adição, subtração, manipulação de bits, ...) sobre os dados.

- Registradores

  Posições de memória construídas dentro da CPU. Muito mais rápido do que acesso a memória principal (RAM).

- UC

  Controla todo o fluxo de execução da CPU. Busca dados e instruções da memória e coordena a troca de dados entre registradores e ULA.

## Funcionamento da CPU

- Tarefas complexas são decompostas em seqüência de instruções simples.
- Cada instrução é executa movendo-se dados dos registradores para a ULA, que executa o cálculo ou operação apropriada e retorna dados para os registradores.
- Ciclo da CPU:
  → Caminho de uma instrução: dos registradores até a ULA e da ULA até os registradores, passando por barramentos.
  → Velocidade do computador é medida em ciclos por segundo.
- Para uma tarefa ser executada, ela deve ser carregada da memória principal (instrução a instrução) na ULA.
- A ULA executa e devolve o resultado para a memória.

**Funcionamento da CPU**

- Tarefa: c = a + b
- Seqüência de execução na CPU:
  - → R0 recebe o valor de a
  - → R1 recebe o valor de b
  - → R0 e R1 são passados para ULA
  - → ULA executa a soma
  - → Resultado é devolvido para R2
  - → Posição de memória associada a c recebe o resultado

**Execução:**

- O programa é carregado para a memória.
- O sistema operacional controla a seqüência das operações e alocação de memória.
- Os programas são lidos a partir da primeira instrução até chegar a uma instrução de stop ou parar.
- Unidade de controle recebe cada uma das instruções, interpreta e dá seqüência a fluxo de execução operativa (registradores – ULA)

**Programação:**

- O conjunto de instruções possíveis que uma CPU pode executar é chamado de linguagem de máquina.
- Programas podem controlar o comportamento do computador, através de instruções em linguagens de máquina.
- Na prática, os programas (conjuntos de instruções em sequência) são programados em linguagens de programação ditas de alto nível e convertidos para linguagem de máquina por compiladores.

## Linguagens de programação:

Conjunto finito de comandos que são combinados (programados) de tal forma a produzir um programa, para realizar uma tarefa.

- Baixo nível – são linguagens de máquina, portanto, é o que o computador consegue interpretar e executar.
- Alto nível – são linguagens que facilitam a programação por parte do programador, pois estão mais próximas da linguagem humana (Fortran, C/C++, Java, Pascal, ...).

## Compilador:

É um programa que converte as instruções de um programa escrito em uma linguagem de alto nível em instruções de máquina.

**Linguagem de Máquina:**

- São comandos simples que a CPU pode executar.
- Podem então ser combinados para produzir uma tarefa mais complexa.
- Exemplos de comandos baixo nível:

  → Registrador R0 recebe um valor

  → Registrador R1 recebe um valor

  → ULA computa operação de soma

  → R2 recebe o resultado da soma feita pela ULA

**Linguagem de Máquina:**

- Na prática, cada instrução da ULA está associada a uma seqüência de bits. Por exemplo:

  → Soma poderia ser: 01001000

  → Atribuição de variável: 00000001

  → Desvio de fluxo: 01001110

- Impossível programar desta forma!

Alternativa:

Uso da linguagem de baixo nível Assembly (segunda geração)

## Linguagem Assembly:

Notação legível por humanos, na forma de mnemônicos, como alternativa à linguagem de máquina. A tradução do código Assembly para a linguagem de máquina é feita pelo montador ou assembler.

```
section     .text
global      _start                          ;must be declared for linker (ld)

_start:                                     ;tell linker entry point

    mov     edx,len                         ;message length
    mov     ecx,msg                         ;message to write
    mov     ebx,1                           ;file descriptor (stdout)
    mov     eax,4                           ;system call number (sys_write)
    int     0x80                            ;call kernel

    mov     eax,1                           ;system call number (sys_exit)
    int     0x80                            ;call kernel

section     .data

msg     db  'Hello, world!',0xa             ;our dear string
len     equ $ - msg                         ;length of our dear string
```

Limitações: Díficil manipulação e dependente da arquitetura do computador.

## Linguagem Assembly:

Notação legível por humanos, na forma de mnemônicos, como alternativa à linguagem de máquina. A tradução do código Assembly para a linguagem de máquina é feita pelo montador ou assembler.

```
section     .text
global      _start                          ;must be declared for linker (ld)

_start:                                     ;tell linker entry point

    mov     edx,len                         ;message length
    mov     ecx,msg                         ;message to write
    mov     ebx,1                           ;file descriptor (stdout)
    mov     eax,4                           ;system call number (sys_write)
    int     0x80                            ;call kernel

    mov     eax,1                           ;system call number (sys_exit)
    int     0x80                            ;call kernel

section     .data

msg     db  'Hello, world!',0xa             ;our dear string
len     equ $ - msg                         ;length of our dear string
```

Solução: Programar em linguagens de alto nível: FORTRAN.

# FORTRAN - Histórico

1954-1957: The IBM Mathematical Formula Translating System

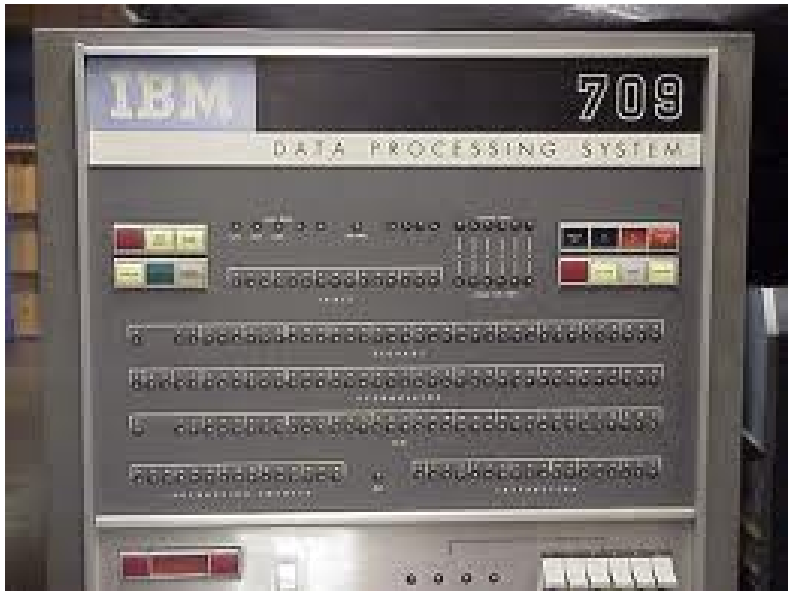
John W. Backus da IBM:

Fortran I: alternativa à linguagem Assembly para a programação do IBM 704.

# FORTRAN - Histórico

1957-1978: Atualizações do FORTRAN disponíveis

- 1958: FORTRAN II e III
- 1961: FORTRAN IV
- 1963: mais de 40 compiladores FORTRAN a disposição
- 1966: FORTRAN 66 como padrão
- 1978: FORTRAN 77




→ Padrão fixo: formatação do código baseada no cartão perfurado

# FORTRAN - Histórico

1957-1978: Atualizações do FORTRAN disponíveis

- 1958: FORTRAN II e III
- 1961: FORTRAN IV
- 1963: mais de 40 compiladores FORTRAN a disposição
- 1966: FORTRAN 66 como padrão
- 1978: FORTRAN 77

→ Padrão fixo: formatação do código baseada no cartão perfurado

# FORTRAN - Histórico

1957-1978: Atualizações do FORTRAN disponíveis

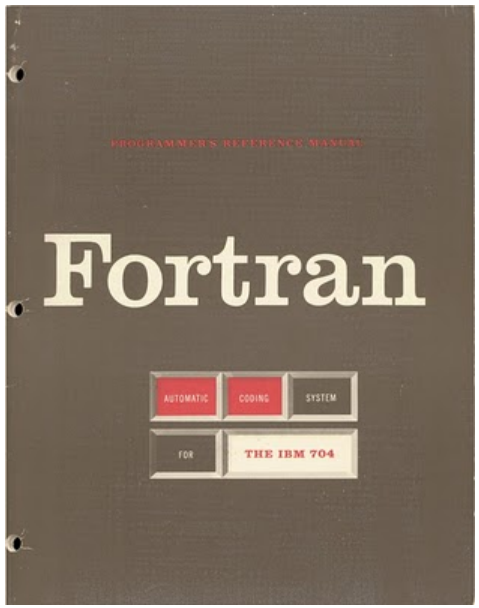
- 1958: FORTRAN II e III
- 1961: FORTRAN IV
- 1963: mais de 40 compiladores FORTRAN a disposição
- 1966: FORTRAN 66 como padrão
- 1978: FORTRAN 77

→ Padrão fixo: formatação do código baseada no cartão perfurado

# FORTRAN - Histórico

1957-1978: Atualizações do FORTRAN disponíveis

- 1958: FORTRAN II e III
- 1961: FORTRAN IV
- 1963: mais de 40 compiladores FORTRAN a disposição
- 1966: FORTRAN 66 como padrão
- 1978: FORTRAN 77

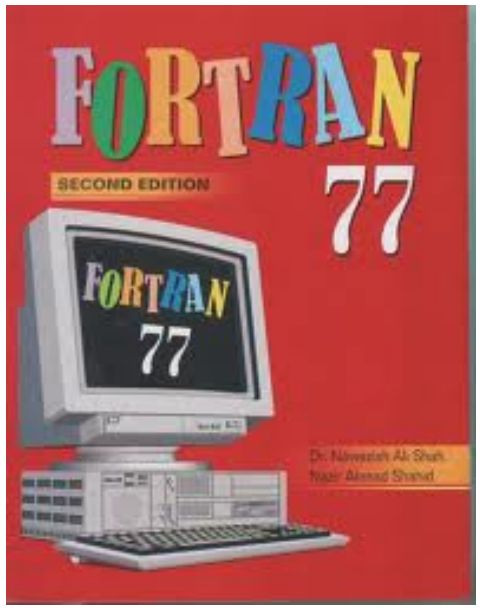



→ Padrão fixo: formatação do código baseada no cartão perfurado

# FORTRAN - Histórico

1957-1978: Atualizações do FORTRAN disponíveis

- 1958: FORTRAN II e III
- 1961: FORTRAN IV
- 1963: mais de 40 compiladores FORTRAN a disposição
- 1966: FORTRAN 66 como padrão
- 1978: FORTRAN 77

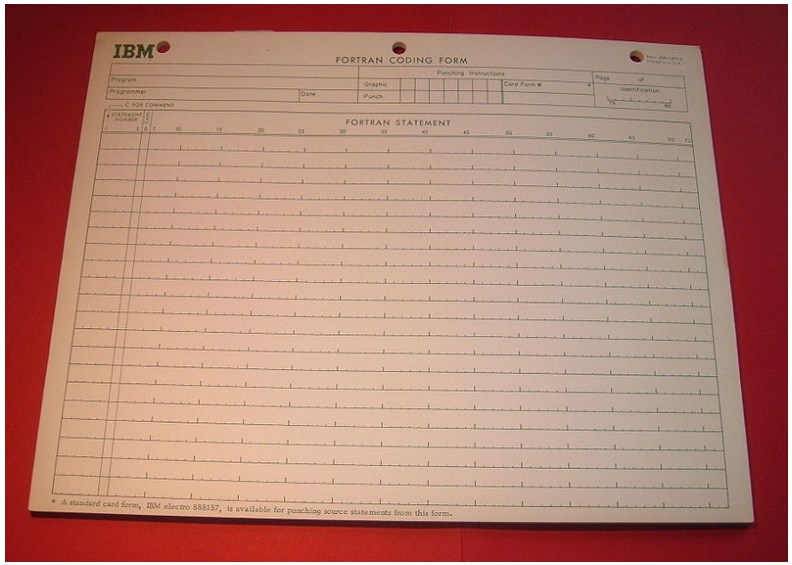



→ Padrão fixo: formatação do código baseada no cartão perfurado

# FORTRAN - Histórico

1957-1978: Atualizações do FORTRAN disponíveis

- 1958: FORTRAN II e III
- 1961: FORTRAN IV
- 1963: mais de 40 compiladores FORTRAN a disposição
- 1966: FORTRAN 66 como padrão
- 1978: FORTRAN 77




→ Padrão fixo: formatação do código baseada no cartão perfurado

# FORTRAN - Histórico

1957-1978: Atualizações do FORTRAN disponíveis

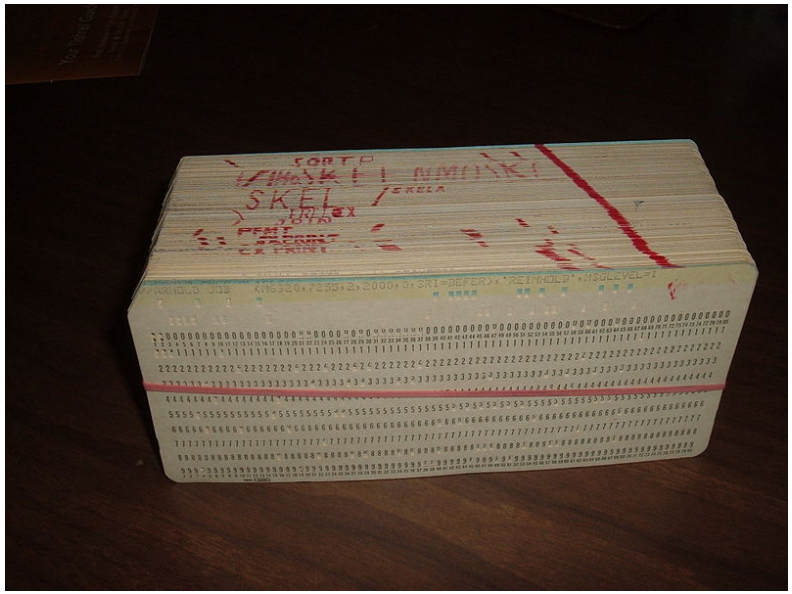
- 1958: FORTRAN II e III
- 1961: FORTRAN IV
- 1963: mais de 40 compiladores FORTRAN a disposição
- 1966: FORTRAN 66 como padrão
- 1978: FORTRAN 77

→ Padrão fixo: formatação do código baseada no cartão perfurado

# FORTRAN - Histórico

1957-1978: Atualizações do FORTRAN disponíveis

- 1958: FORTRAN II e III
- 1961: FORTRAN IV
- 1963: mais de 40 compiladores FORTRAN a disposição
- 1966: FORTRAN 66 como padrão
- 1978: FORTRAN 77



→ Padrão fixo: formatação do código baseada no cartão perfurado

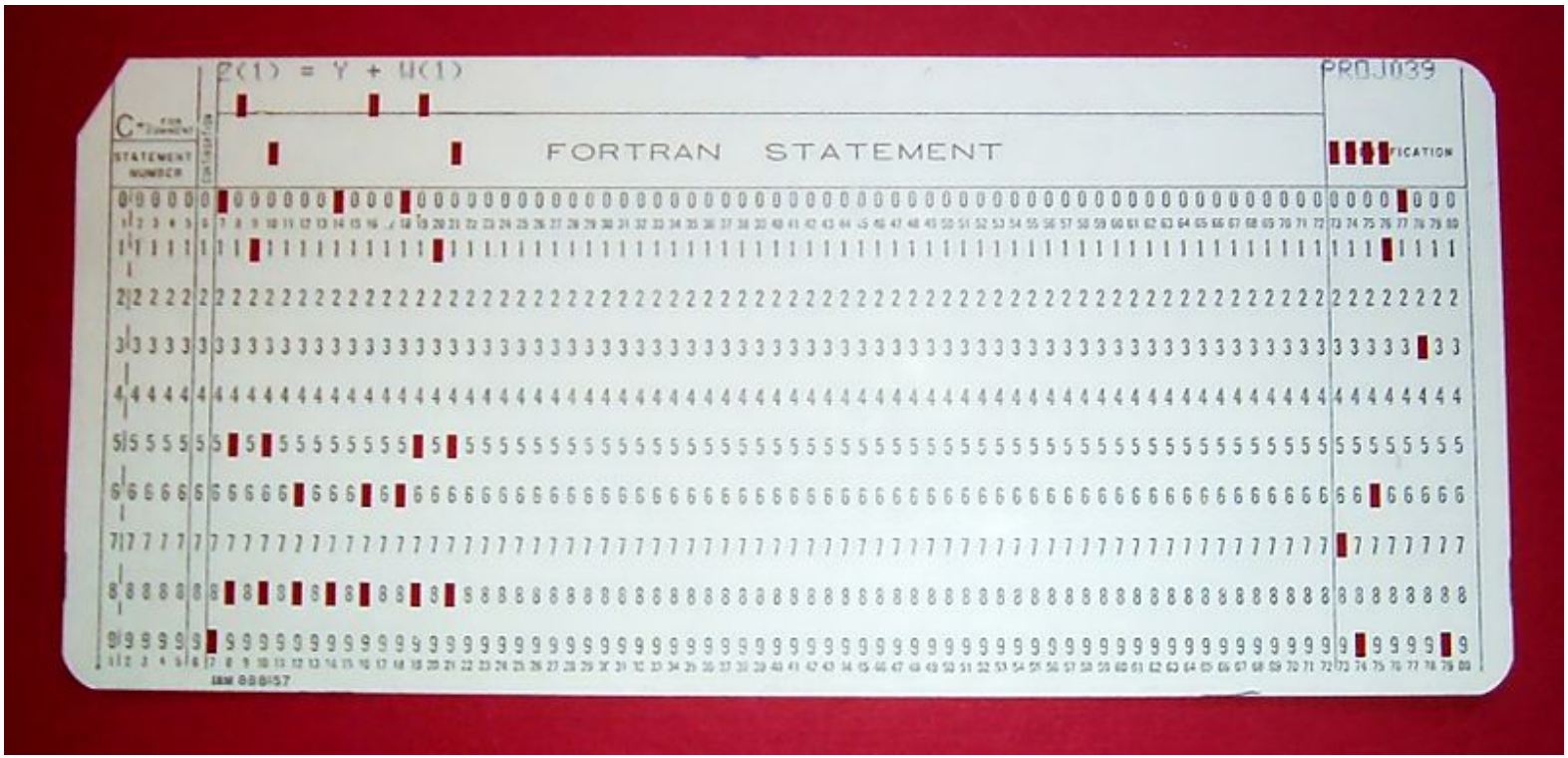
# FORTRAN - Histórico

1957-1978: Atualizações do FORTRAN disponíveis

- 1958: FORTRAN II e III
- 1961: FORTRAN IV
- 1963: mais de 40 compiladores FORTRAN a disposição
- 1966: FORTRAN 66 como padrão
- 1978: FORTRAN 77

→ Padrão fixo: formatação do código baseada no cartão perfurado

1957-1978: Estrutura de um programa em FORTRAN



1957-1978: Estrutura de um programa em FORTRAN

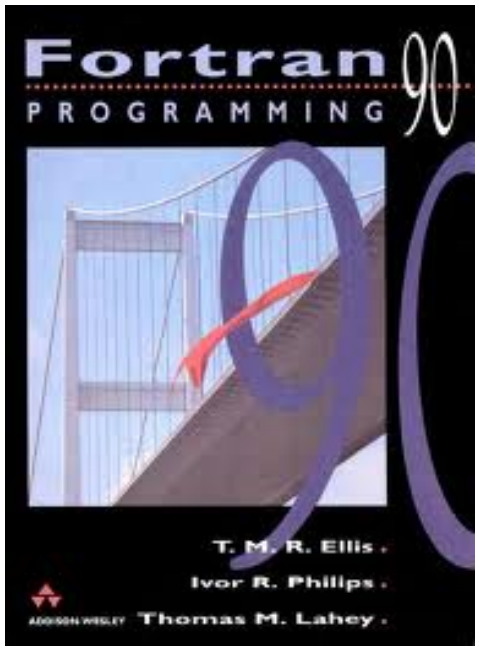
```
C FORTRAN IV program to
C find sum of integers 1-100
      INTEGER I, SUM
      SUM=0
      DO 1 I=1, 100
    1 SUM = SUM + I
      WRITE (6,2) SUM
    2 FORMAT(1X,I5)
      STOP
      END
```

FIGURE 114. FORTRAN IV program

# FORTRAN - Histórico

1991-hoje: uma nova versão para o Fortran

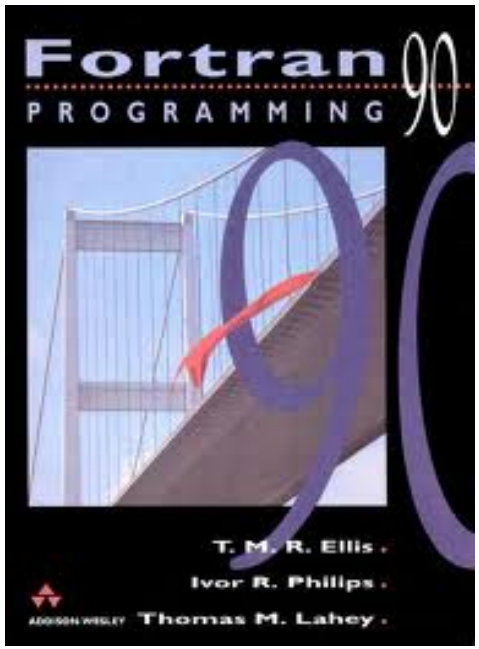
- 1992: Fortran 90
  Fim da formatação do cartão!
- 1997: Fortran 95
  Fortran de alta performance.
- 2004: Fortran 2003
  Na direção da programação orientada à objeto.
- 2010: Fortran 2008
  Atualização do Fortran 2003.



# FORTRAN - Histórico

1991-hoje: uma nova versão para o Fortran

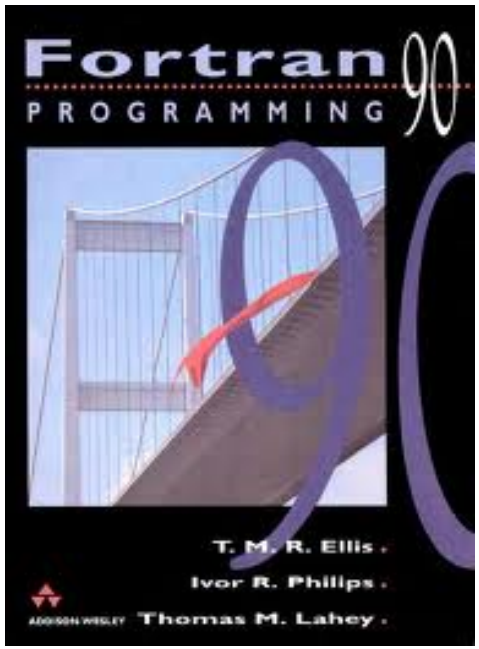
- 1992: Fortran 90
  Fim da formatação do cartão!
- 1997: Fortran 95
  Fortran de alta performance.
- 2004: Fortran 2003
  Na direção da programação orientada à objeto.
- 2010: Fortran 2008
  Atualização do Fortran 2003.



# FORTRAN - Histórico

1991-hoje: uma nova versão para o Fortran

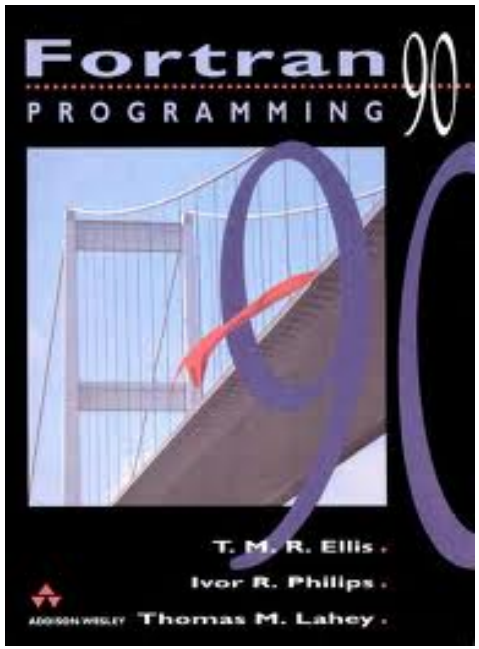
- 1992: Fortran 90
  Fim da formatação do cartão!
- 1997: Fortran 95
  Fortran de alta performance.
- 2004: Fortran 2003
  Na direção da programação orientada à objeto.
- 2010: Fortran 2008
  Atualização do Fortran 2003.



# FORTRAN - Histórico

1991-hoje: uma nova versão para o Fortran

- 1992: Fortran 90
  Fim da formatação do cartão!
- 1997: Fortran 95
  Fortran de alta performance.
- 2004: Fortran 2003
  Na direção da programação orientada à objeto.
- 2010: Fortran 2008
  Atualização do Fortran 2003.



# FORTRAN - Histórico

## Compiladores FORTRAN disponíveis:

- GNU gfortran (https://gcc.gnu.org/wiki/GFortran)
- Intel ifort (https://software.intel.com/en-us/intel-compilers)
- g95 (http://www.g95.org/)
- The Portland Group (http://www.pgroup.com/)
- NAGWare (http://www.fortran.com/fortran/nag.html)

## Mais Opções:

https://www.fortran.com/compilers.html