A CPU EXECUTA OPERAÇÕES PRIMITIVAS: SOMA, SUBTRAÇÃO, AND, OR, MOVER UM DADO DE UMA POSIÇÃO PARA OUTRA, TRANSFERIR DADO DE/PARA DISPOSITIVOS DE I/O, ETC.

CADA OPERAÇÃO DEVE ESTAR PRECISAMENTE REPRESENTADA (CODIFICADA) EM UMA SEQUÊNCIA DE 0's E 1's.

UMA INSTRUÇÃO DE MÁQUINA DITA EXATAMENTE UMA OPERAÇÃO QUE A CPU DEVE EXECUTAR. ELA É COMPOSTA BASICAMENTE DO CÓDIGO DA OPERAÇÃO E UM GRUPO DE BITS QUE PERMITE LOCALIZAR OS OPERANDOS ENVOLVIDOS NAQUELA OPERAÇÃO (CASO HAJA OPERANDOS).

UM PROGRAMA DE COMPUTADOR É PORTANTO UM CONJUNTO DE INSTRUÇÕES DE MÁQUINA, SEQUENCIALMENTE ORGANIZADO.

PARA QUE UM PROGRAMA POSSA SER EXECUTADO ELE DEVE ESTAR NA MEMÓRIA PRINCIPAL E O ENDEREÇO DA PRIMEIRA INSTRUÇÃO DEVE ESTAR NA CPU PARA QUE A MESMA POSSA BUSCÁ-LA E DAR INÍCIO A EXECUÇÃO.

A FUNÇÃO BÁSICA DE UMA CPU É ENTÃO:

- BUSCAR UMA INSTRUÇÃO NA MEMÓRIA (EFETUAR UMA LEITURA);

- INTERPRETAR QUAL OPERAÇÃO AQUELA INSTRUÇÃO ESTÁ EXPLICITANDO;

- BUSCAR OS DADOS ENVOLVIDOS NA OPERAÇÃO;

- EXECUTAR EFETIVAMENTE A OPERAÇÃO, GUARDANDO O RESULTADO (SE HOUVER) NO LOCAL ESPECIFICADO.

REINICIAR O PROCESSO DE BUSCA PARA A PRÓXIMA INSTRUÇÃO.

ATUANDO NO AUXÍLIO A CPU ESTÃO:

A UAL (UNIDADE ARITIMÉTICA E LÓGICA), QUE É O DISPOSITIVO QUE REALMENTE EXECUTA AS OPERAÇÕES ARITMÉTICAS E LÓGICAS: SOMA, SUB, MULT, DIV, AND, OR, XOR, OP COMPLEMENTO, DESLOCAMENTO (DIR, ESQ) INCREMENTA DECREMENTA

E OS REGISTRADORES, QUE SÃO MEMÓRIAS ESPECÍFICAS DA CPU CUJA QUANTIDADE E TIPOS DEPENDEM DA ARQUITETURA DA MÁQUINA EM QUESTÃO (ARQUITETURAS RISC X SISC). OS REGISTRADORES MAIS COMUNS ENTRE AS ARQUITETURAS EXISTENTES SÃO:

ACC (ACUMULADOR): REGISTRADOR ENVOLVIDO NA MAIORIA DAS OPERAÇÕES ARITMÉTICAS.

PSW (PROGRAM STATUS WORD): REGISTRADOR DE ESTADO DA CPU, SEU CONTEUDO ESTÁ RELACIONADO COM:

- OCORRÊNCIA OU NÃO DE OVERFLOW, CARRY, PARIDADE.

-OCORRÊNCIA OU NÃO DE OVERFLOW, CARRY, PARIDADE;

- MÁSCARAS

- MODO DE OPERAÇÃO (SUPER OU USER);

- ETC.

RI (REGISTRADOR DE INSTRUÇÃO): CONTÉM SEMPRE A INSTRUÇÃO QUE ESTÁ SENDO EXECUTADA; CI OU PC (PROGRAM COUNTER) É O CONTADOR DE INSTRUÇÃO. CONTÉM SEMPRE O ENDEREÇO DA PRÓXIMA INSTRUÇÃO A SER EXECUTADA; (PARA QUE UM PROGRAMA POSSA SER EXECUTADO, O ENDERÇO DA SUA PRIMEIRA INSTRUÇÃO É CARREGADO NO PC)

REM (REGISTRADOR DE ENDEREÇO DE MEMÓRIA) CONTÉM O ENDEREÇO DE MEMÓRIA ENVOLVIDO EM UMA OPERAÇÃO DE LEITURA/GRAVAÇÃO.

RDM (REGISTRADOR DE DADOS DA MEMÓRIA) CONTÉM O DADO ENVOLVIDO NA ÚLTIMA OPERAÇÃO DE LEITURA/GRAVAÇÃO.

A INSTRUÇÃO DE MÁQUINA AINDA SEGUE A ARQUITETURA BÁSICA PROPOSTA POR JOHN von NEUMANN QUE É CALCADA ESSENCIALMENTE NA EXISTÊNCIA DE UMA SEQUÊNCIA DE ORDEM OU INSTRUÇÃO QUE FAZ COM QUE O HARDWARE REALIZE UMA DETERMINADA TAREFA. (Também se caracteriza pela possibilidade de uma máquina digital armazenar seus programas no mesmo espaço de memória que os dados, podendo assim manipular tais dados)

A ARQUITETURA DE von NEUMANN É PORTANTO PRÓPRIA PARA PROGRAMAS ESCRITOS EM LINGUAGENS IMPERATIVAS.

CICLO DE INSTRUÇÃO:

FORMATO DE INSTRUÇÃO:

COMO JÁ VISTO AS INSTRUÇÕES DE MÁQUINA DIFEREM LARGARMENTE DE ARQUITETURA PARA ARQUITETURA, MAS, BASICAMENTE SÃO FORMADAS POR DOIS ELEMENTOS: O CÓDIGO DA OPERAÇÃO E O ELEMENTO QUE INDICA PARA CPU, COM QUE DADOS AQUELA OPERAÇÃO IRÁ SE REALIZAR (OS OPERANDOS).

SEUS FORMATOS E TAMANHOS SÃO MUITO VARIÁVEIS. POR EXEMPLO SE UM DETERMINADO PROCESSADOR RESERVAR 6 BITS PARA O CÓDIGO DA OPERAÇÃO, ENTÃO ESSE PROCESSADOR PODERÁ REALIZAR APENAS 64 INSTRUÇÕES DIFERENTES.

QUANTO AOS OPERANDOS, DOIS ASPECTOS PODEM SER ANALISADOS:

QUANTIDADE DE OPERANDOS (1, 2, 3, 4)

MODO DE INTERPRETAR O VALOR ARMAZENADO NO CAMPO OPERANDO, CONHECIDO COMO MODO DE ENDEREÇAMENTO.

MODOS DE ENDEREÇAMENTO:

IMEDIATO - O VALOR DO CAMPO CORRESPONDENTE AO OPERANDO JÁ É O PRÓPRIO DADO.

DIRETO – O VALOR DO CAMPO É O ENDEREÇO DE MEMÓRIA ONDE ESTÁ O DADO.

INDIRETO – O VALOR DO CAMPO É O ENDEREÇO DE MEMÓRIA ONDE ESTÁ O ENDEREÇO DEFINITIVO DO DADO.

HÁ AINDA DOIS MÉTODOS DE INDICAR O MODO DE ENDEREÇAMENTO DE UMA INSTRUÇÃO:

IMPLÍCITO: A INSTRUÇÃO CONTÉM NO SEU CÓDIGO TANTO A OPERAÇÃO A SER REALIZADA, COMO O MODO DE ENDEREÇAMENTO.

EXPLÍCITO: A INSTRUÇÃO CONTÉM UM CAMPO ESPECÍFICO PARA INDICAR O MODO DE ENDEREÇAMENTO. NESSE CAMPO CONSTA UM CÓDIGO BINÁRIO CORRESPONDENTE AO MODO DESEJADO.

IMPLÍCITO

CÓDIGO DA OPERAÇÃO	OPERANDO

A OPERAÇÃO LOAD PODERIA TER ENTÃO 3 CÓDIGOS DIFERENTES:

LDI ACC <<< OPERANDO (END. IMEDIATO)

LDA ACC <<< (OPERANDO) (END. DIRETO)

LDID ACC <<< ((OPERANDO))

(END. INDIRETO)

EXPLÍCITO

A OPERAÇÃO LOAD TERIA APENAS UM CÓDIGO, POR EXEMPLO: LDA

O MODO DE ENDEREÇAMENTO COM DOIS BITS SERIA, POR EXEMPLO:

00 - DIRETO

01 - IMEDIATO

10 - INDIRETO

11 - NÃO UTILIZADO

MODELO DE COMPUTADOR

PRINCIPAIS ELEMENTOS:

CPU

BUS (canal de comunicação)

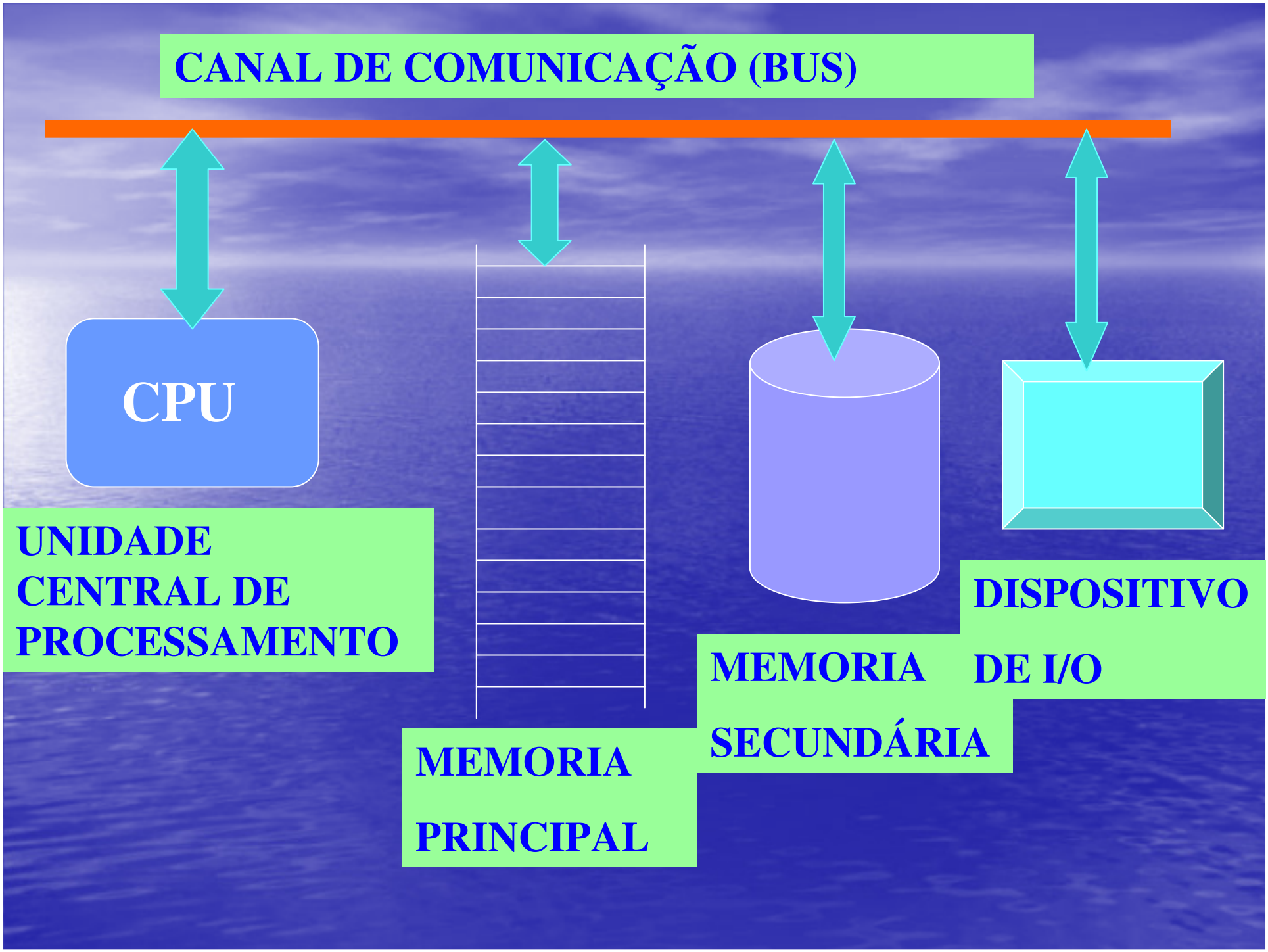
MEMÓRIA PRINCIPAL

(acesso randômico e armazenamento volátil)

MEMÓRIA SECUNDÁRIA

(meio magnético: HD, não volátil, custo e velocidade de acesso baixos)

DISPOSITIVOS DE ENTRADA E SAÍDA


CANAL DE COMUNICAÇÃO (BUS)
CPU
UNIDADE CENTRAL DE PROCESSAMENTO
MEMORIA PRINCIPAL
MEMORIA SECUNDÁRIA
DISPOSITIVO DE I/O

UM COMPUTADOR HIPOTÉTICO

CPU POSSUI UMA ÁREA DE MEMORIA INTERNA CAPAZ DE ARMAZENAR, TEMPORARIAMENTE, O RESULTADO DE UMA OPERAÇÃO (REGISTRADOR)

CADA POSIÇÃO DE MEMÓRIA PODE ARMAZENAR VALORES NUMÉRICOS E É ROTULADA POR UM NÚMERO INTEIRO (0,1, ,N)

PROGRAMA: SEQUÊNCIA DE INTRUÇÕES ARMAZENADA NA MEMORIA

INSTRUÇÕES DISPONÍVEIS

(MNEMÔNICOS, ASSEMBLY):

READ pos : captura o valor fornecido via teclado e o armazena na posição "pos";

WRITE pos : escreve o valor armazenado na posição de memória rotulada por "pos" na tela do computador;

STORECONST num pos :armazena o valor da constante numérica especificada por "num" na posição de memória rotulada por "pos";

ADD pos1 pos2 : calcula a soma dos dois operandos que são os valores numéricos armazenados nas posições “pos1” e “pos2”.

O resultado é armazenado no REGISTRADOR.

SUB pos1 pos2 : calcula a diferença entre o primeiro e segundo operando, o resultado é armazenado no REGISTRADOR.

MUL pos1 pos2 : multiplica o primeiro e segundo operando, o resultado é armazenado no REGISTRADOR.

DIV pos1 pos2 : divide o primeiro pelo segundo operando, o resultado é armazenado no registrador.

LOAD pos : carrega o registrador com o conteúdo da memória da posição “pos”.

STORE pos : armazena o valor do registrador na posição de memória rotulada por “pos”.

PROGRAMA EM LINGUAGEM DE

ALTO NÍVEL:

```
INT   X ;
READ ( X );
X = X + 1;
WRITE ( X );
```

COMPILADOR

VAR	POS
X	0
C01	1

PROGRAMA EM LINGUAGEM ASSEMBLY

GERADO PELO COMPILADOR

ASSEMBLY É UMA LINGUAGEM DE PROGRAMAÇÃO?

```
INT   X ;
READ ( X );
X = X + 1;
WRITE ( X );
```

```
READ X
STORECONST  1 C01
ADD X C01
STORE X
WRITE X
```

COMPILADOR

VAR	POS
X	0
C01	1

GERAÇÃO DE CÓDIGO:

- MNEMÔNICOS SUBSTITUIDOS PELOS CÓDIGOS DAS RESPECTIVAS INSTRUÇÕES
- NOME DAS VARIÁVEIS E CONSTANTES SUBSTITUIDOS PELOS RESPECTIVOS ENDEREÇOS DE MEMÓRIA

NO NOSSO COMPUTADOR HIPOTÉTICO VAMOS SUPOR OS SEGUINTES CÓDIGOS :

INSTRUÇÃO	CÓDIGO
READ	01
WRITE	02
STORECONST	03
ADD	04
SUB	05
MUL	06
DIV	07
STORE	08

```
INT   X ;
READ ( X );
X = X + 1;
WRITE ( X );
```

Programa fonte

Linguagem Assembly

```
READ X
STORECONST  1 C01
ADD X C01
STORE X
WRITE X
```

Assembler ou montador

```
01 0
03 1 1
04 0 1
08 0
02 0
```

Programa em linguagem de máquina

VAR	POS
X	0
C01	1

VOLTEMOS AOS MNEMÔNICOS MAS COM POSIÇÕES REAIS DE MEMÓRIA:

```
READ   0

STORECONST  35.5  1

ADD   0  1

STORE   2

WRITE   2
```

QUAL O VALOR ESCRITO SE O VALOR FORNECIDO FOR 20?

```
READ   0
STORECONST  35.5  1
ADD   0  1
STORE   0
WRITE   0
```

```
READ   0
STORECONST  35.5  1
ADD   0  1
STORE   0
WRITE   2
```


READ 0
ADD 0 1
STORE 2
WRITE 2

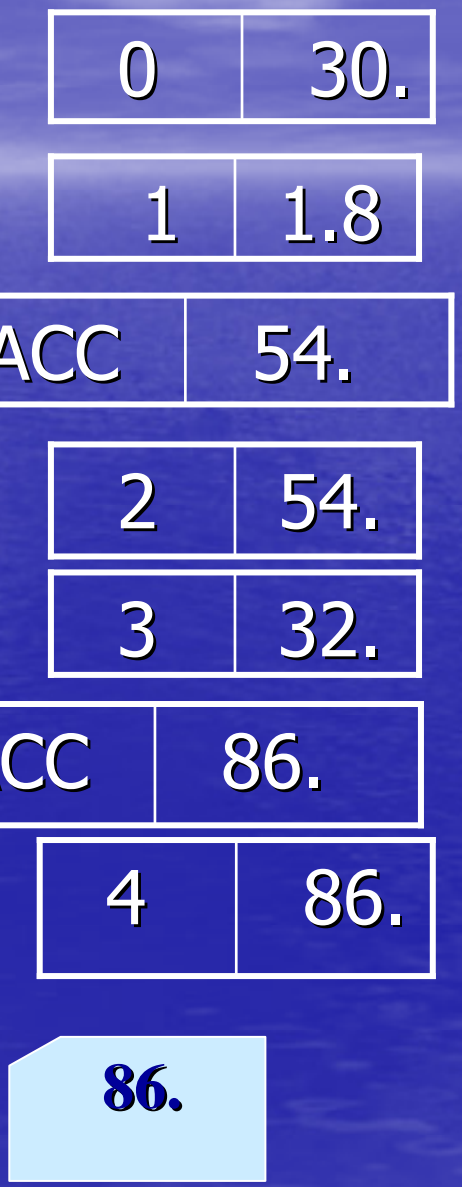
UTILIZANDO NOSSA MÁQUINA HIPOTÉTICA VAMOS CONSTRUIR UM PROGRAMA PARA CONVERTER VALORES DE UMA UNIDADE PARA OUTRA.

O NOSSO PROGRAMA RECEBE UM VALOR NUMÉRICO QUE CORRESPONDE A UMA TEMPERATURA EM GRAUS CELSIUS E EXIBE A TEMPERATURA CORRESPONDENTE EM GRAUS FAHRENHEIT.

A FÓRMULA DE CONVERSÃO ENTRE AS UNIDADES É: $F = 1.8\,C + 32.0$

PROGRAMA:

```
READ    0
STORECONST  1.8    1
MUL       1    0
STORE    2
STORECONST  32    3
ADD        2    3
STORE    4
 WRITE    4
```



CONSTRUIR UM PROGRAMA QUE LEIA TRÊS NOTAS DE UM ALUNO (N1, N2, N3);

CALCULE A MÉDIA FINAL DESSE ALUNO (MF);

ESCREVA ESSA MÉDIA.

AS NOTAS TÊM PESOS DIFERENTES:

N1 PESO 2, N2 PESO 3 E N3 PESO 5

COM AS INSTRUÇÕES VISTA ATÉ AQUI A NOSSA MÁQUINA HIPOTÉTICA É SEQUENCIAL E É CAPAZ DE IMPLEMENTAR APENAS PROGRAMAS QUE APLICAM DIRETAMENTE FÓRMULAS MATEMÁTICAS (FORTRAN John Warner Backus 1954).

MUITOS PROBLEMAS MAIS COMPLEXOS PODEM SER RESOLVIDOS DE FORMA INCREMENTAL: REALIZANDO CÁLCULOS SIMPLES, REPETIDAMENTE.

O QUE ESTARIA FALTANDO PARA MELHORAR A NOSSA MÁQUINA?

COMO FAZER NOSSA MÁQUINA EXECUTAR REPETIDAMENTE UM TRECHO DE INSTRUÇÕES?

ADICIONANDO A ELA UMA NOVA INSTRUÇÃO:

JUMP pos offset

SALTE PARA A INSTRUÇÃO X SE O CONTEÚDO DE pos FOR MAIOR QUE ZERO.

ENDEREÇO DE X = ENDEREÇO ATUAL + offset

O QUE FAZ O PROGRAMA ABAIXO?

```
READ    0

STORECONST   1   1

WRITE   0

SUB   0    1

STORE   0

JUMP  0  -3
```

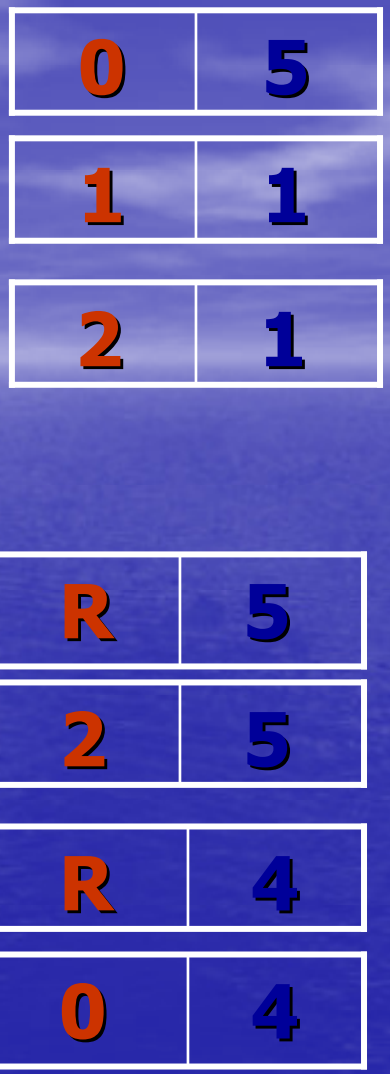
VAMOS AGORA CONSTRUIR UM PROGRAMA QUE FAÇA NOSSA MAQUININHA CALCULAR O FATORIAL DE UM NUMÉRO:

SE N = 0 ENTÃO FAT (N) = 1

SENÃO FAT (N) = N x (N-1) x (N-2)x

(N-3) x ... 3x2x1

```
READ    0
STORECONST  1    1
STORECONST  1    2
JUMP    1    5
MUL     2    0
STORE   2
SUB     0    1
STORE   0
JUMP    0   -4
WRITE   2
```



0	4
1	1
2	5

```
JUMP   1    5
MUL    2    0
STORE  2
SUB    0    1
STORE  0
JUMP   0   -4
WRITE  2
```

R	20
2	20

R	3
0	3

```
JUMP   1    5
MUL    2    0
STORE  2
SUB    0    1
STORE  0
JUMP   0   -4
WRITE  2
```

0	3
1	1
2	20

R	60
2	60

R	2
0	2