EP3

FDX - Primeiro semestre 2015
MAC 0329

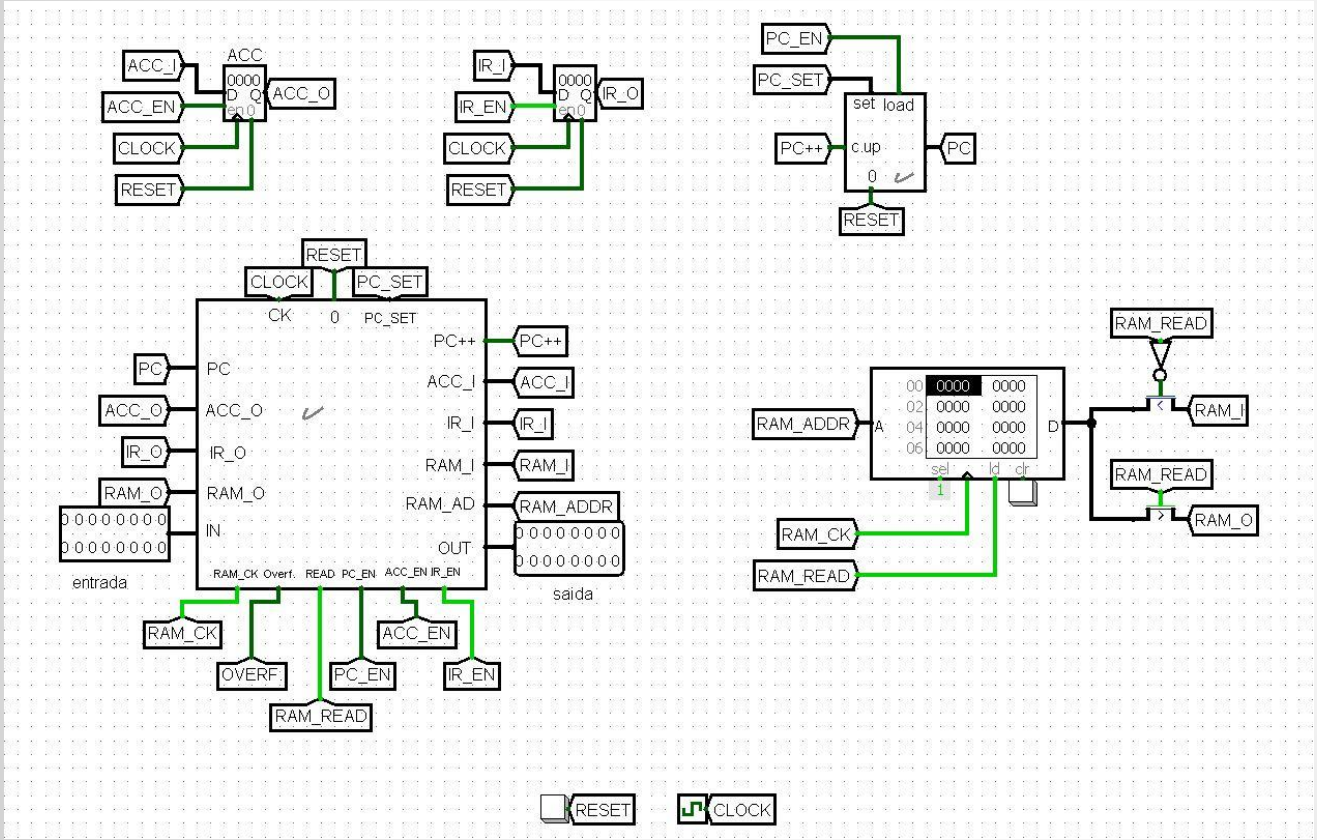
integrantes:

Antonio Augusto Abello	8536152
Leonardo Daneu Lopes	8516816
Lucas Sung Jun Hong	8124329
William Shinji Numada	7648325

Acabamento da HIPO

1. Acoplar ALU (16-bits) na arquitetura HIPO;
2. Implementar desvios condicionais;
3. Tratamento da Entrada/Saída.

Main



Teremos a descrição de cada componente nesta ordem:

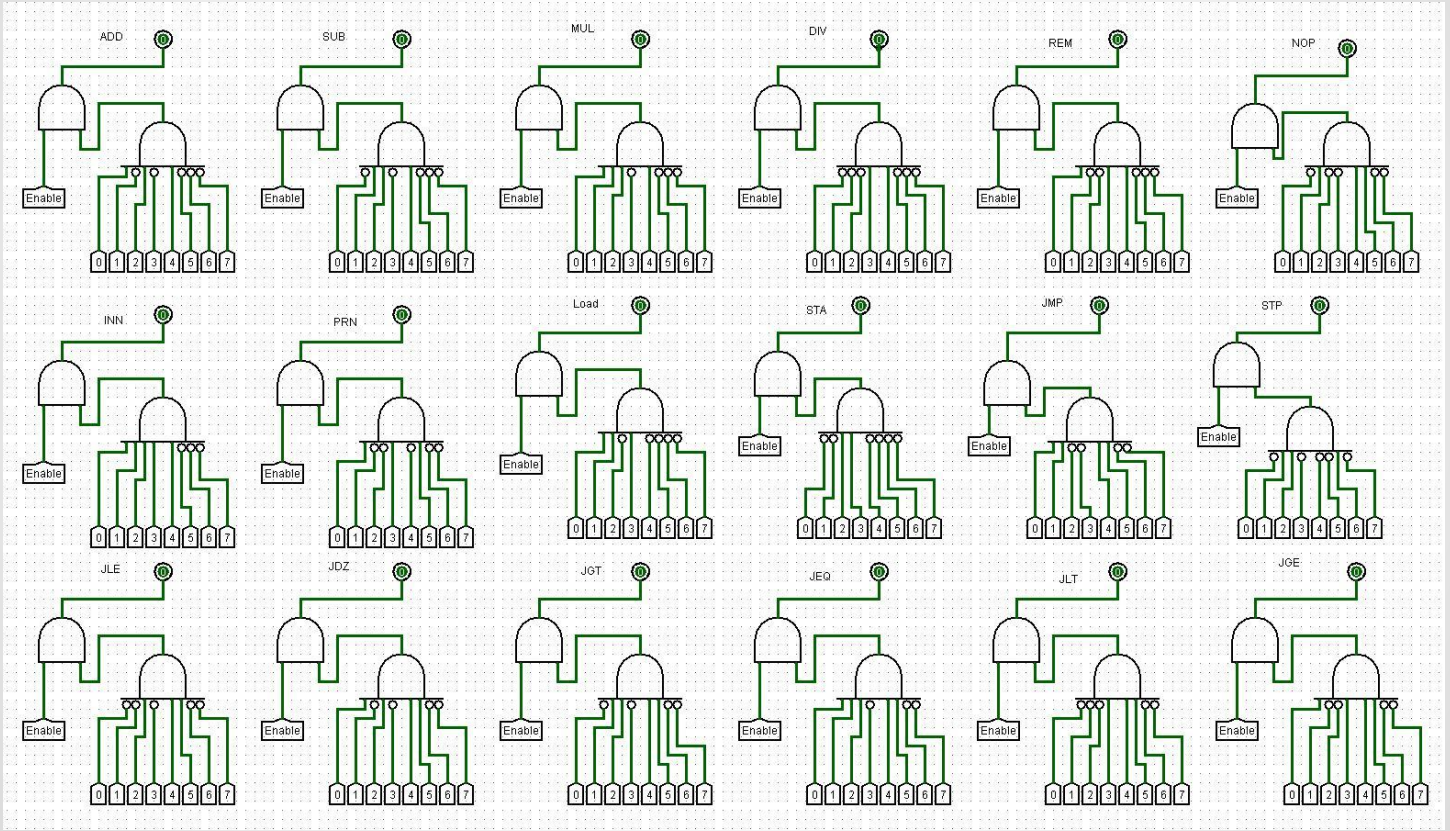
- Acoplamento da ALU;
- Desvios condicionais;
- Entrada/Saída;

Acoplamento da ALU

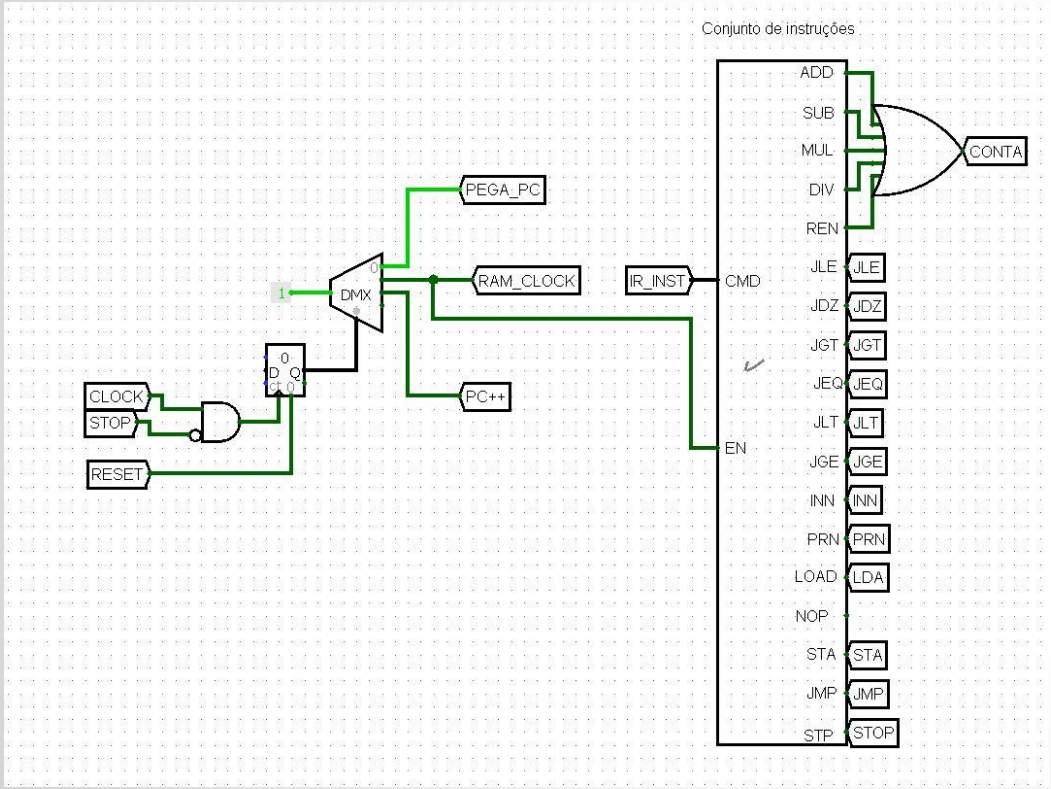
- Adicionamos as novas instruções no conjunto de instruções;

  - JLE: 0X34;
  - JDZ: 0X35;
  - JGT: 0X36;
  - JEQ: 0X37;
  - JLT: 0X38;
  - JGE: 0X39;
  - ADD: 0X15;
  - SUB: 0X16;
  - MUL: 0X17;
  - DIV: 0X18;
  - REM: 0X19;
  - Load: 0X0B;
  - STA: 0X0C;
  - NOP: 0X32;
  - JMP: 0X33;
  - STP: 0X46;
  - INN: 0X1F;
  - PRN: 0X29;

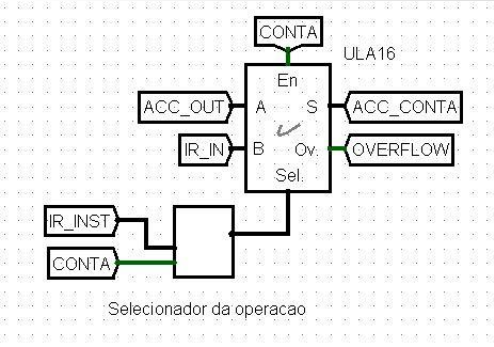
IF (Instruction-fetch ou *if* (condição “se”))



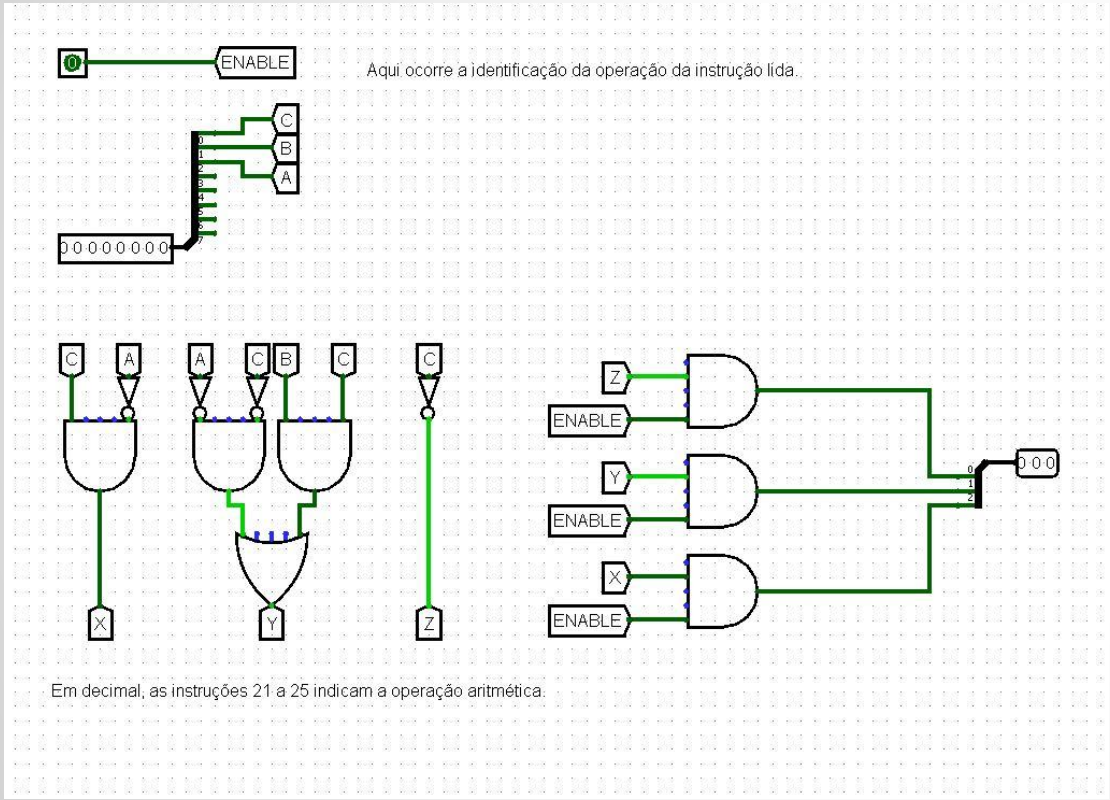
Acoplamento da ALU



- Acoplamos esse novo conjunto de instruções no controlador;
- E finalmente tratamos o ULA no controlador. Quando CONTA estiver ativado, indica que devemos realizar a operação com o dado lido em IR com o dado armazenado no Acumulador. (figura abaixo)

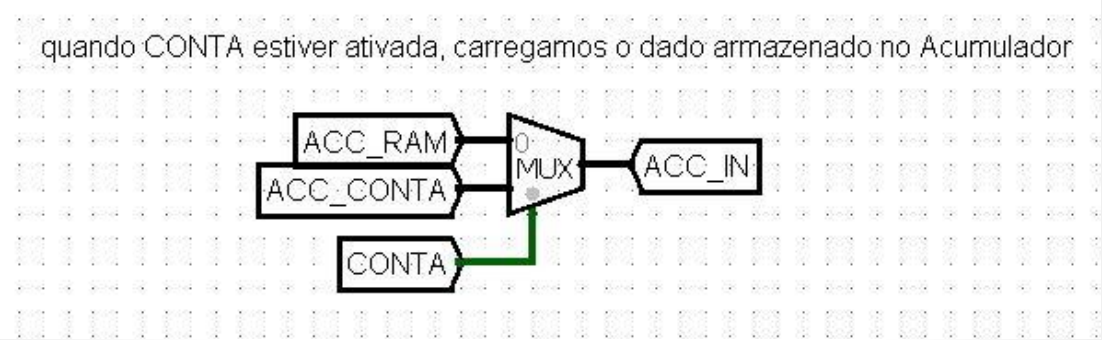


Acoplamento da ALU



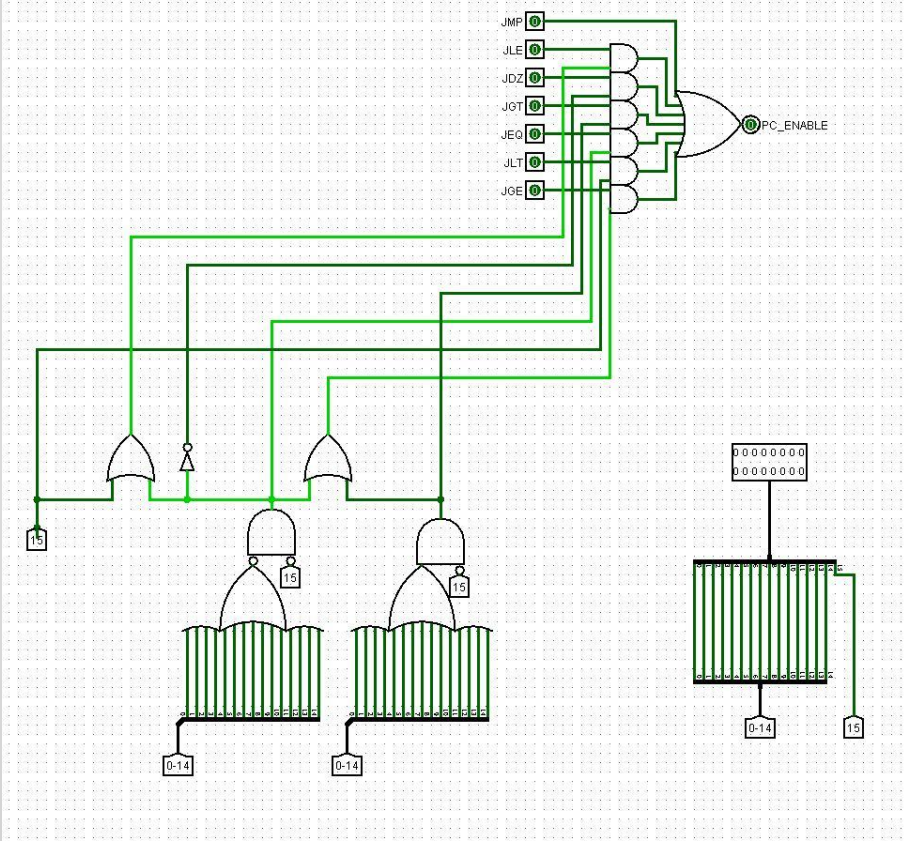
- Quando a instrução for identificada como uma operação aritmética, ENABLE se ativará. Assim, necessita-se identificar se a instrução trata-se de soma, subtração etc;
- Quando instrução (em decimal) for:
  - 21: teremos 001 (SOMA);
  - 22: 010 (SUBTRAÇÃO);
  - 23: 011 (MULTIPLICAÇÃO);
  - 24: 100 (DIVISÃO);
  - 25: 101 (RESTO).
- Portanto a saída desse circuito será usada nos multiplexers na ALU.

Acoplamento da ALU



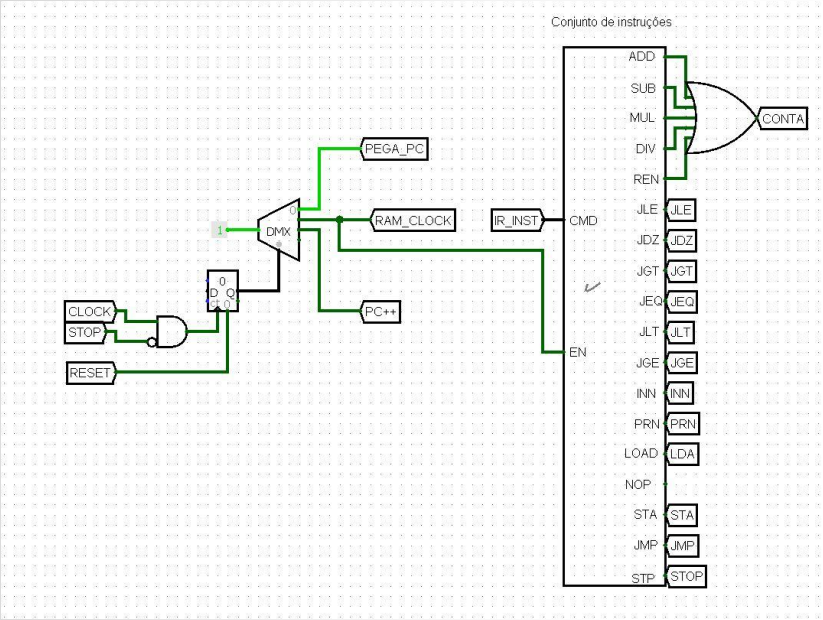
- ACC_IN armazenará o valor do resultado realizado, caso alguma operação aritmética tenha sido solicitada;
- Caso contrário, armazenará apenas um dado quando a instrução tenha sido 0X0B.

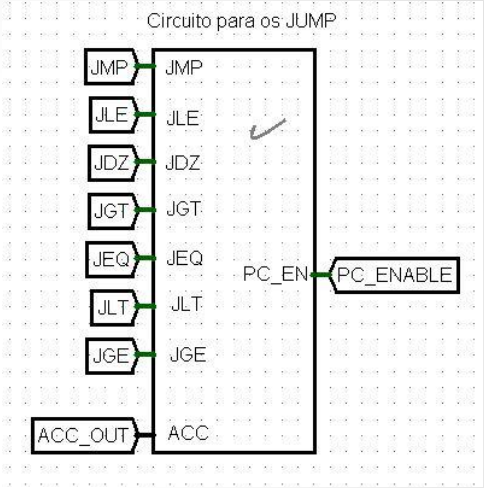
Desvios condicionais



- Temos aqui um analisador do acumulador, que indica caso o acumulador possui um valor:
  - MENOR que zero;
  - MENOR ou IGUAL a zero;
  - IGUAL a zero;
  - DIFERENTE de zero;
  - MAIOR que zero;
  - MAIOR ou IGUAL a zero;
- PC_ENABLE indicará caso uma das condições acima foi satisfeita, também verificando se houve um salto não-condicional, JMP nesse caso.

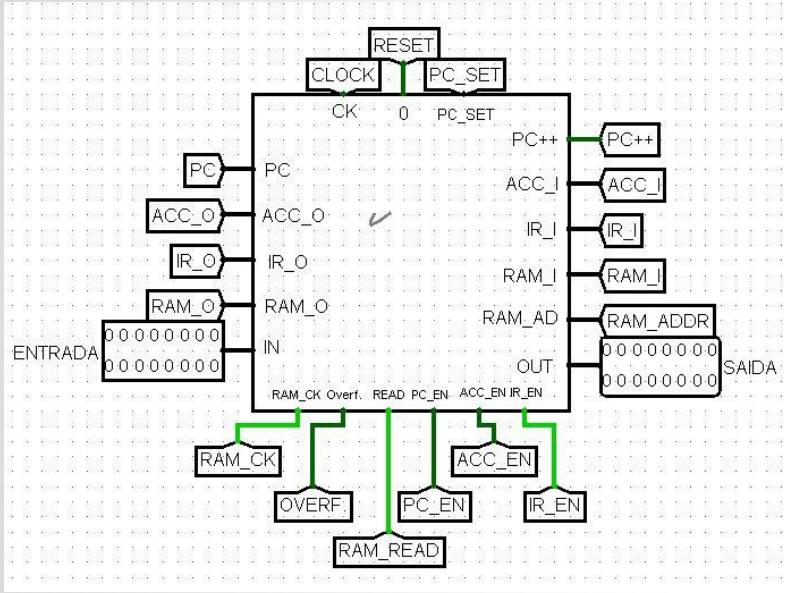
Desvios condicionais

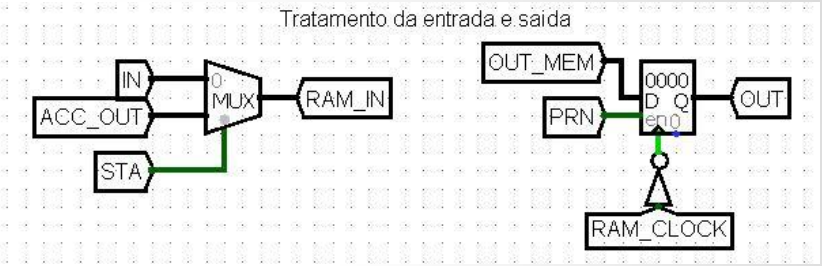




Assim, após a identificação do desvio condicional, verificaremos se o dado armazenado no acumulador satisfaz umas das condições de desvio, ativando PC_ENABLE.

Entrada / Saída





- No controlador, temos os componentes para a entrada e saída;
- Como no conjunto de instruções decodifica-se se a instrução lida se trata de 0x1F (INN) ou 0X29 (PRN), a figura acima mostra que se instrução for:
  - INN: o endereço passa a ter o dado do acumulador;
  - PRN: dado do endereço será passado para a saída.

EXEMPLO

EXEMPLO

endereço	instrução
00	1f0c
01	0b0c
02	3407
03	180d
04	3407
05	0c0e
06	290e
07	4600
.	
.	
.	
0c	0000
0d	0e10
0e	0000

Algoritmo criado:

Conversor de segundos para horas. O programa recebe, em ENTRADA, um número binário X segundos e converte para H horas em binário, imprimindo o resultado em SAIDA.

Pseudo-código:

- Recebe ENTRADA e armazena X no endereço 0c;
- Verifica se X > 0. Caso sim, continue. Caso contrário, termine programa e imprima 0;
- Divide X / 3600 = H;
- Verfica se H > 0. Caso sim, continue. Caso contrário, termine programa e imprima 0;
- Armazena H no endereço 0e e imprima esse resultado para SAIDA.

EXEMPLO

endereço	instrução
00	1f0c
01	0b0c
02	3407
03	180d
04	3407
05	0c0e
06	290e
07	4600
.	
.	
.	
0c	0000
0d	0e10
0e	0000

Step-by-step:

Endereço 00:

- 1f0c: Temos a instrução INN, então lemos da ENTRADA o número inserido em binário pelo usuário;
- Guardamos esse número no endereço 0c;
- Consideremos esse número inserido como X;

Endereço 01:

- 0b0c: Armazenamos o dado do endereço 0c no ACC (Acumulador);

Endereço 02:

- 3407: Verificamos se o número guardado no ACC é MENOR ou IGUAL a zero;
- Caso JLE (0x34), pulamos para o endereço 07, que indica STOP;

Endereço 03:

- 180d: Dividimos o valor do Acumulador com o dado do endereço 0d (0e10 = 3600 em binário);
- O resultado ficará armazenado no Acumulador;

EXEMPLO

endereço	instrução
00	1f0c
01	0b0c
02	3407
03	180d
04	3407
05	0c0e
06	290e
07	4600
.	
.	
.	
0c	0000
0d	0e10
0e	0000

Endereço 04:

- 3407: Novamente verificamos se resultado X / 3600 <= 0;
- Caso sim, teremos 0 horas e programa termina;
- Caso contrário, prosseguimos para a impressão desse número;

Endereço 05:

- 0c0e: Carregamos o ACC no endereço 0e;

Endereço 06:

- 290e: Temos aqui a instrução PRN, então imprimimos o dado do endereço 0e para a SAIDA;

Endereço 07:

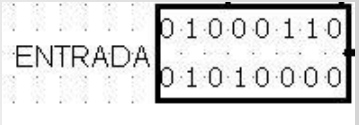
- 4600: Parada do programa.

Endereço 0c: guardará resultado X;

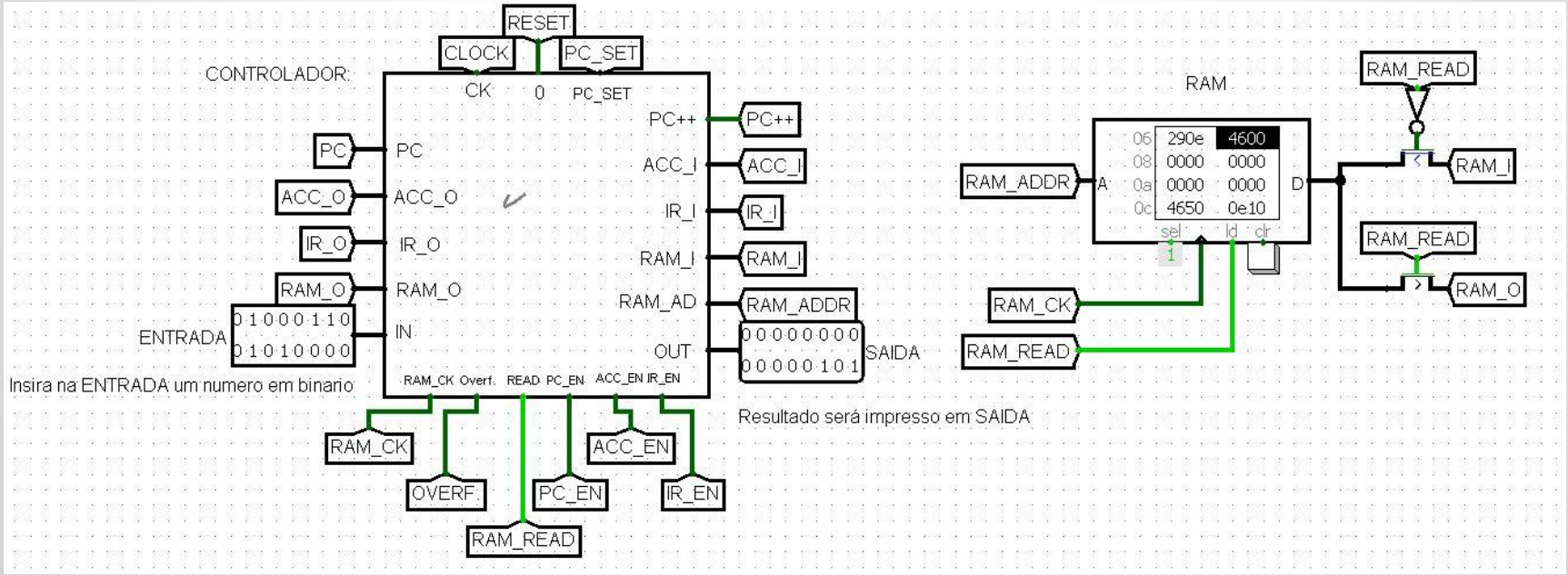
Endereço 0d: possui o valor 3600;

Endereço 0e: guardará resultador H = X / 3600;

EXEMPLO



- Testamos com o número 18000 em decimal (4650 em hex e 0100011001010000 em binário);
- 18000 segundos equivale a 5 horas;
- Então esperamos que a saída seja 0000000000000101;



endereço	instrução
00	1f07
01	0b07
02	3609
03	3705
04	4600
05	2908
06	4600
07	0000
08	0001
09	1608
0a	0c0b
0b	0000
0c	0b07
0d	170b
0e	3610
0f	4600
10	0c07
11	0b0b
12	2907
13	3302

EXEMPLO

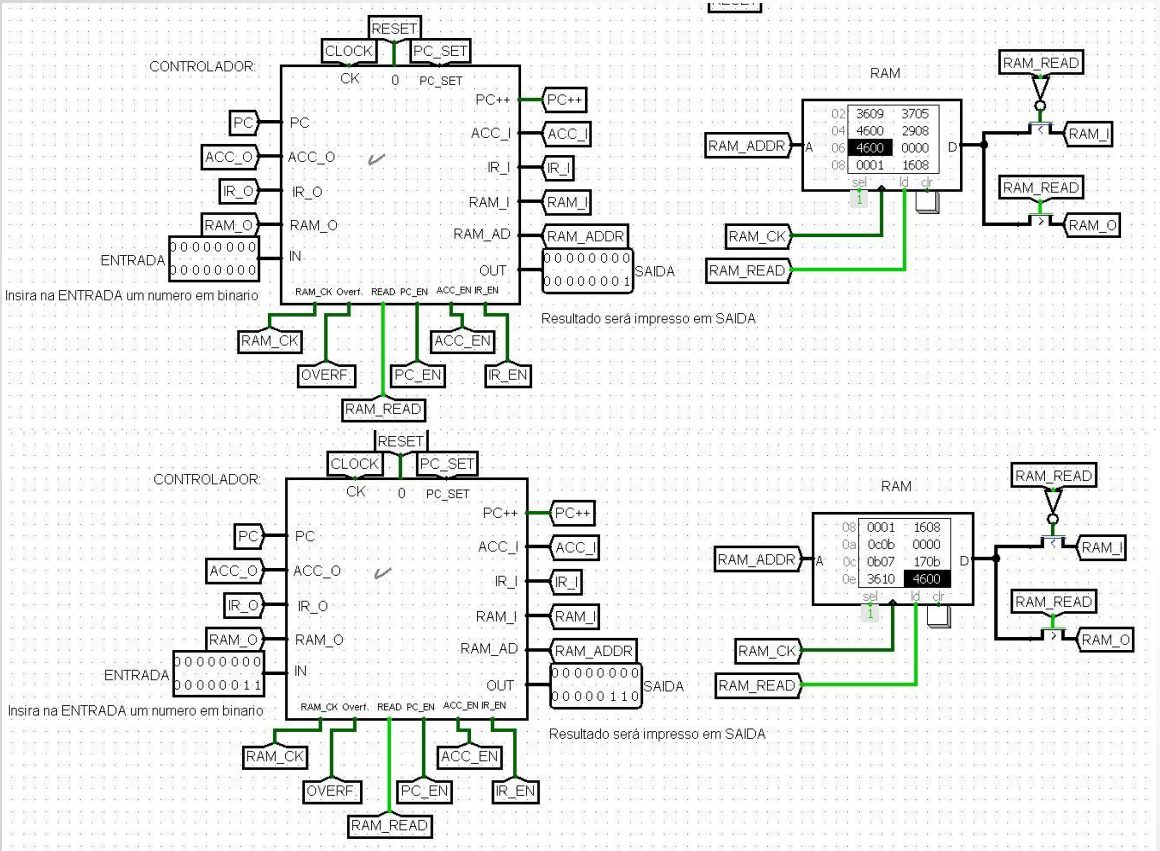
Algoritmo criado:

Função fatorial. O programa recebe, em ENTRADA, um número binário X, imprimindo o resultado (X!) em SAIDA.

Pseudo-código:

- Recebe ENTRADA e armazena X no endereço 0c;
- Verifica se X > 0. Caso sim, continue;
- Verifica se X = 0, imprime resultado 1 e termina programa;
- Se X < 0, imprime 0 e termina programa;
- Endereço 0b, que funcionará como variável A que recebe valor de X - 1 no início e vai decrementando por 1;
- Enquanto A for diferente de 0:
    - X = X * A; (o valor de X atualizado será armazenado no endereço 07)
    - A --;
    - Atualiza o valor de SAIDA;
- Se A = 0, termina programa.

EXEMPLO



- Quando ENTRADA é 0000000000000000, a SAIDA do programa é 1.
- Quando ENTRADA é 0011 (3 em decimal), a SAIDA do programa é 0110 (6 em decimal).