EP2

HIPO - Primeiro semestre 2015
MAC 0329

integrantes:

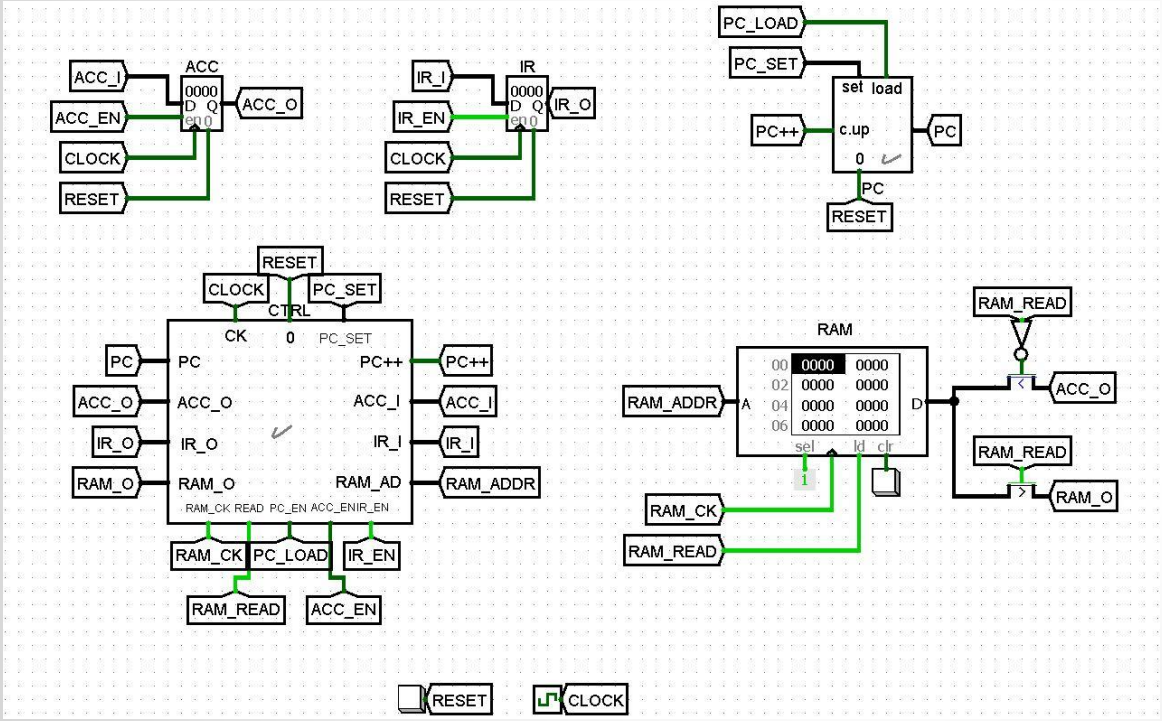
Antonio Augusto Abello	8536152
Leonardo Daneu Lopes	8516816
Lucas Sung Jun Hong	8124329
William Shinji Numada	7648325

Implementação do ciclo de instrução do computador HIPO

etapas do ciclo de instruções:

1. leitura do par instrução/endereço apontado por PC (Program Counter), da memória, armazenando em IR (instruction register);
2. PC++;
3. Decodificação da instrução armazenada no IR;
4. Execução da instrução decodificada;
5. Se instrução executada != STOP, retornar para o item 1.

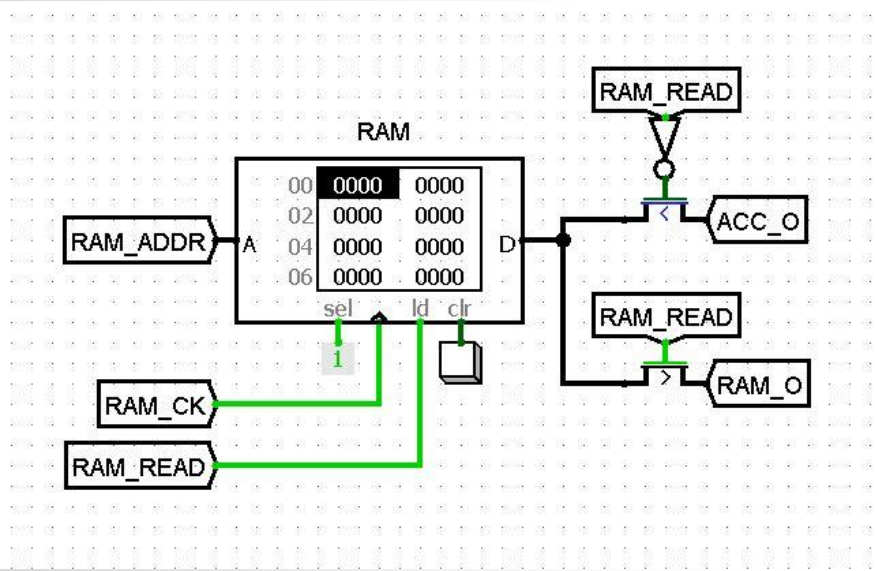
Main



Nos slides seguintes teremos a descrição de cada componente. Será dada a explicação nesta ordem:

- Memória;
- Acumulador / IR;
- PC;
- Controlador:
  - main;
  - IF (conjunto de instruções);
  - parte fetch-decode;
  - parte execute;

Memória

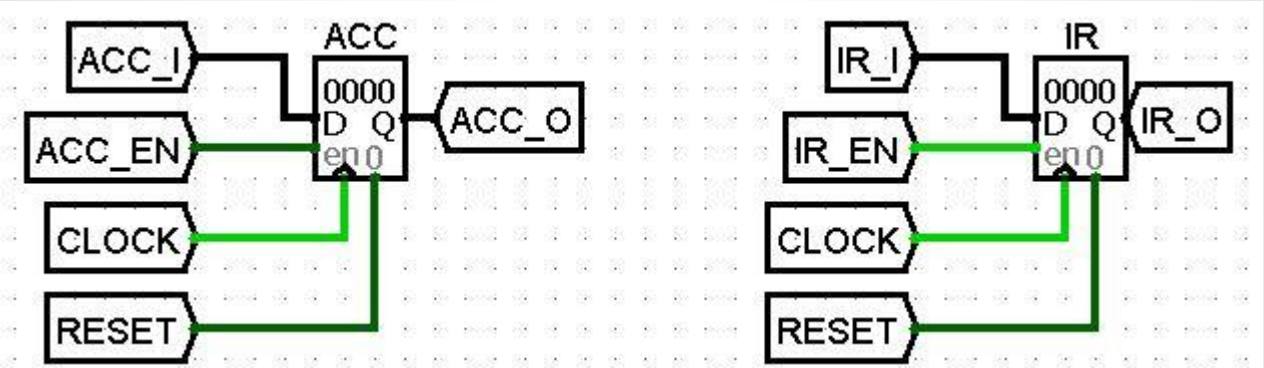


- Usamos RAM (Random Access Memory) do Logisim, com unidades de 16 bits, com endereçamento de 8 bits.

- RAM_ADDR corresponde ao local acessado na memória

- RAM_READ ativa-se quando iniciamos a leitura da instrução em um endereço:
    - RAM_O: dado armazenado na memória;
    - ACC_O: dado carregado no acumulador

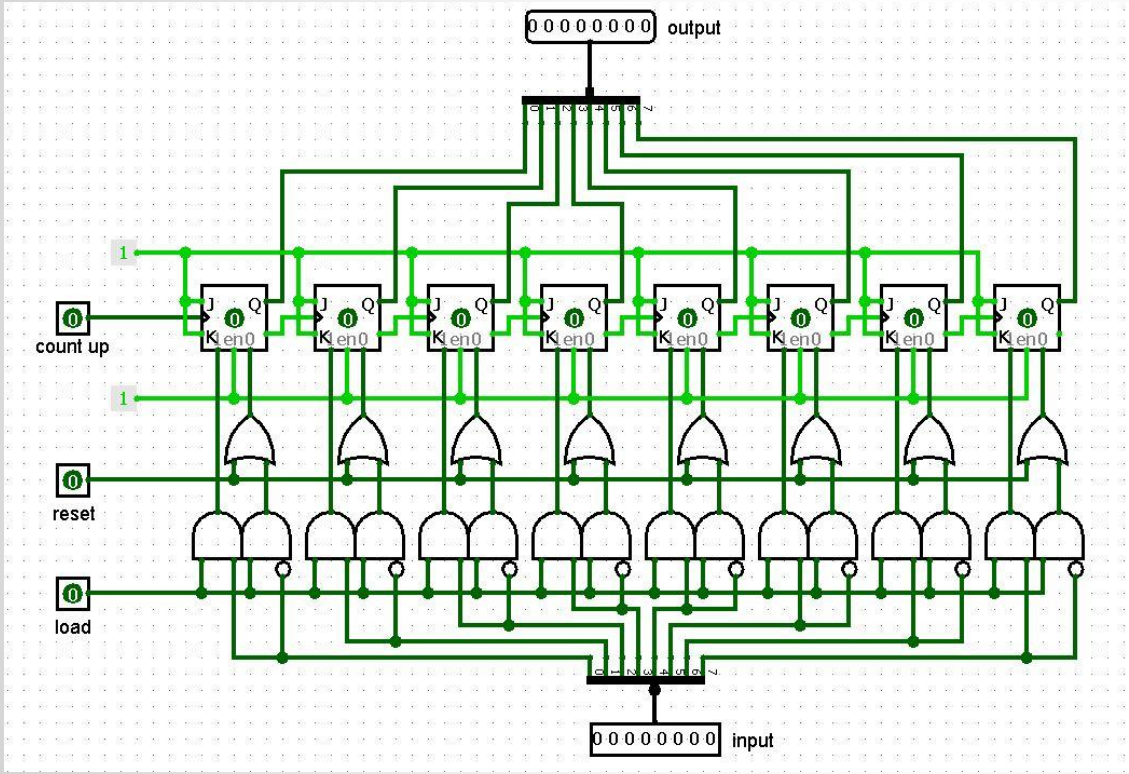
Acumulador / IR



ACC (Acumulador) e IR (Instruction Register)

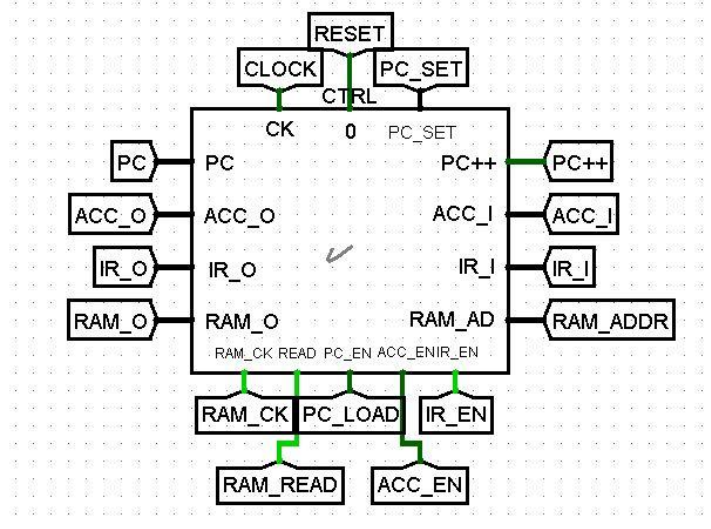
- Usamos os registradores do Logisim;
- Ambos com 16 bits;
- ACC armazena um dado;
- IR armazena:
  - no primeiro byte um código de instrução;
  - no segundo byte um endereço de memória.

PC (Program Counter)



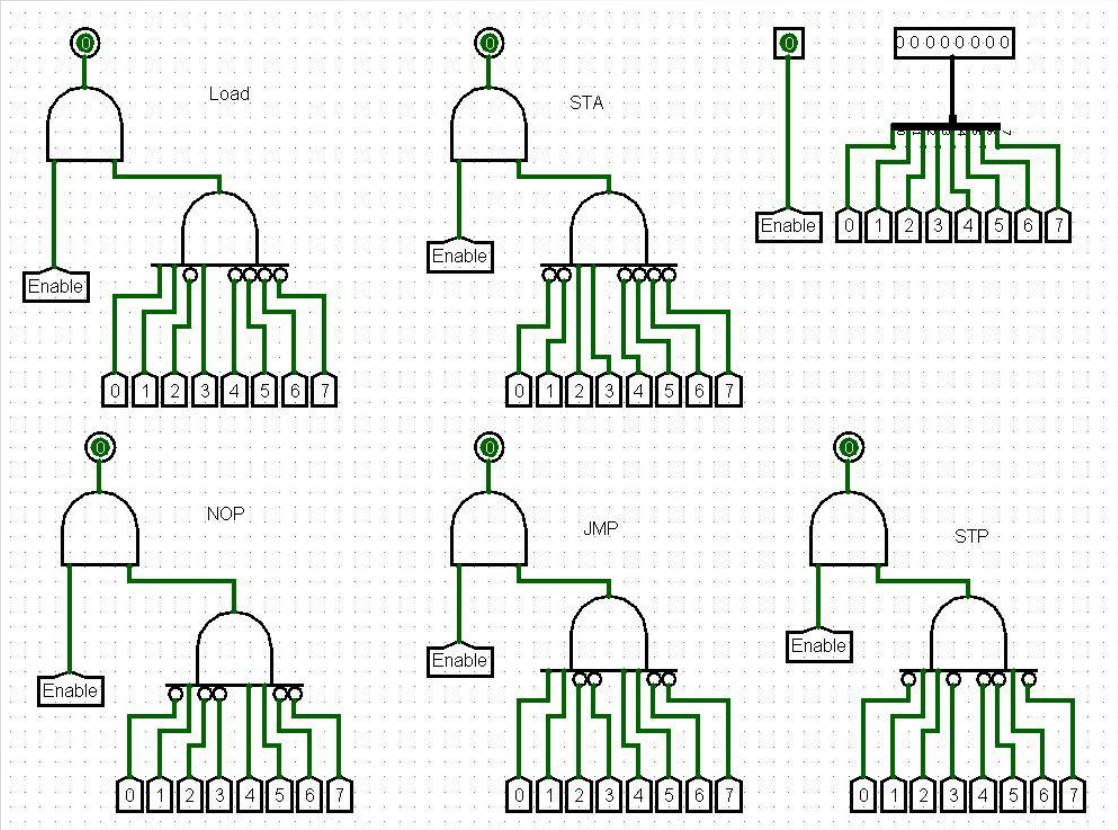
- Implementação com Flip-Flops JK;
- Count up permite a incrementação do PC a cada “pulso”;
- reset permite que PC aponte para 0x00;
- load permite que PC armazene o endereço de memória.

Controlador (main)



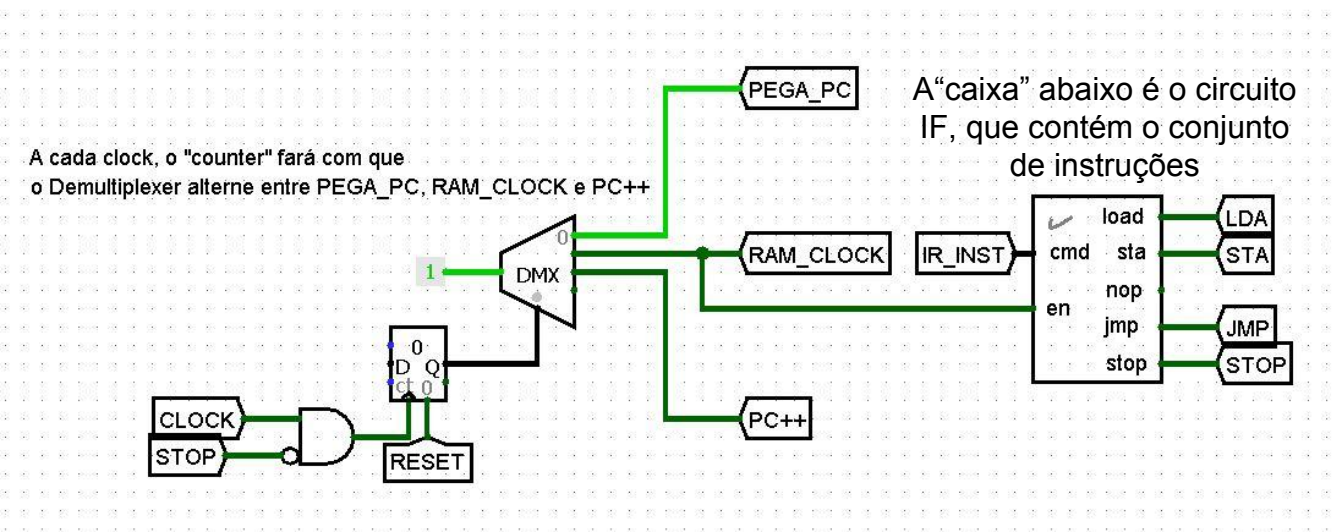
- Encarregado para decodificação e execução de uma instrução armazenada em IR, incremento do PC e definição do modo de operação;

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



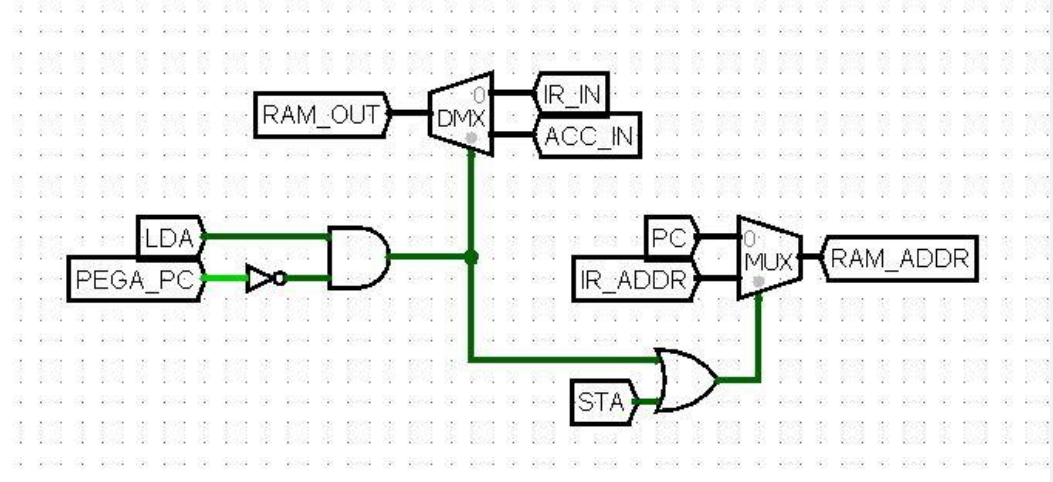
- Implementação do conjunto de instruções;
  - Load: 0x0B;
  - STA: 0x0C;
  - NOP: 0x32;
  - JMP: 0x33;
  - STP: 0x46;
- O circuito do topo-direito da figura é a entrada ou a instrução lida de IR;
- Esse circuito se encontra acoplado com o controlador.

Controlador (parte fetch-decode)



- Quando a instrução lida indicar STOP, o counter não será ativado, cessando qualquer atividade;
- A etapa “fetch” será quando PEGA_PC for ativado;
- A etapa “decode” será quando RAM_CLOCK estiver ativado. Lemos a instrução armazenada em IR e decodificaremos ela, decidindo que instrução devemos realizar no conjunto de instruções;
- A etapa “execute” depende a etapa decode;

Controlador (parte execute)



- A etapa “execute” depende a etapa decode;
- Quando LDA (operação 0x0B) estiver ativado, o dado lido será carregado no acumulador;
- Quando STA (operação 0x0C) estiver ativado, o acumulador será carregado no endereço lido;

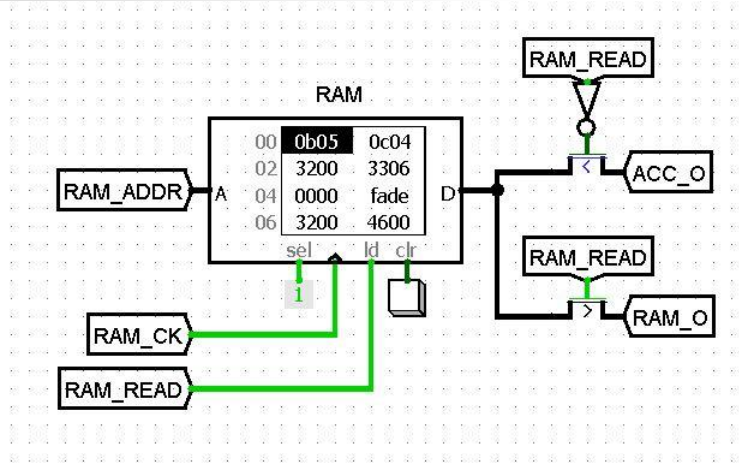
EXEMPLO

EXEMPLO

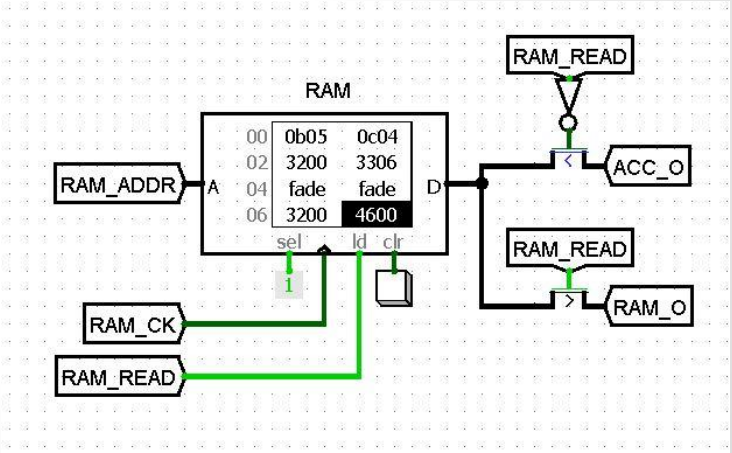
Objetivo:

- Ler um dado chamado “fade”, que está no endereço 05;
- Executar esse dado no endereço 04;
- Terminar o ciclo.

Então teríamos dessa situação:



Para essa:

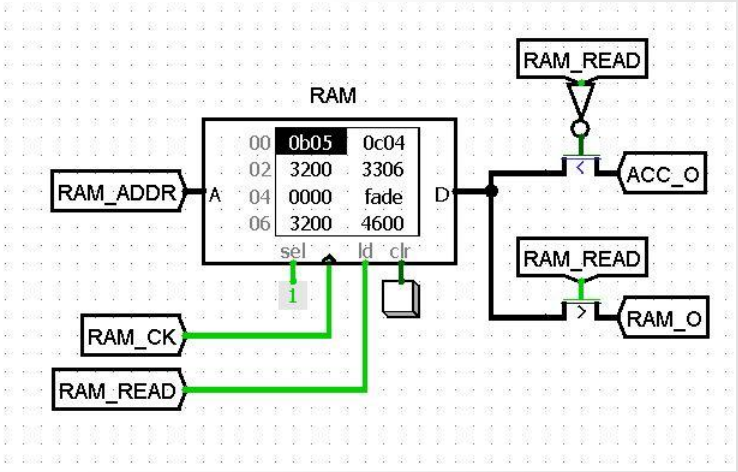


EXEMPLO

Precisamos então da seguinte lista de instruções:

endereço:	instrução:
00	0b05
01	0c04
02	3200
03	3306
04	0000
05	fade
06	3200
07	4600

Em Logisim:

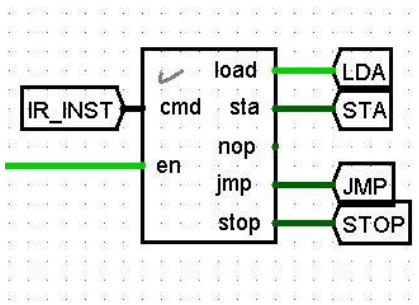


Primeiro clock

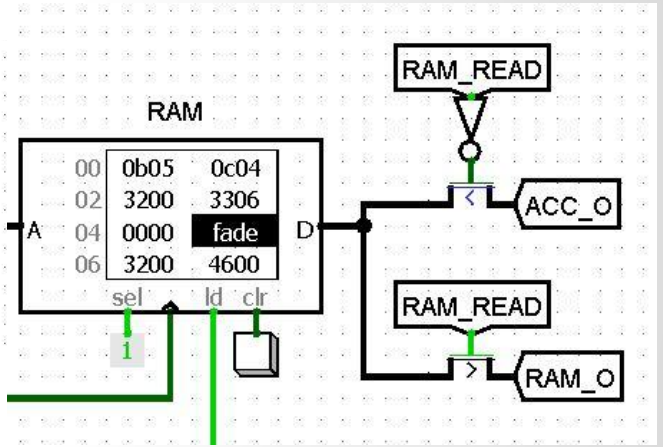
- Antes do primeiro clock, o dado do endereço 00 (0b05) foi armazenado em IR;
- No primeiro clock, teremos o carregamento do dado do endereço 05 no Acumulador;

endereço:	instrução:
00	0b05
01	0c04
02	3200
03	3306
04	0000
05	fade
06	3200
07	4600

IR armazena 0b05



Instrução 0x0B indica carregar o endereço no Acumulador



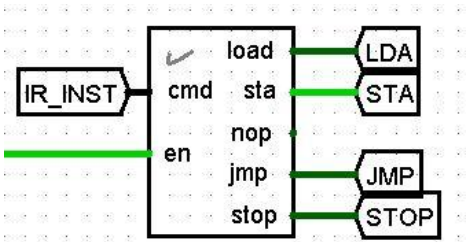
endereço 05 contém instrução chamada: “fade”

Quarto clock

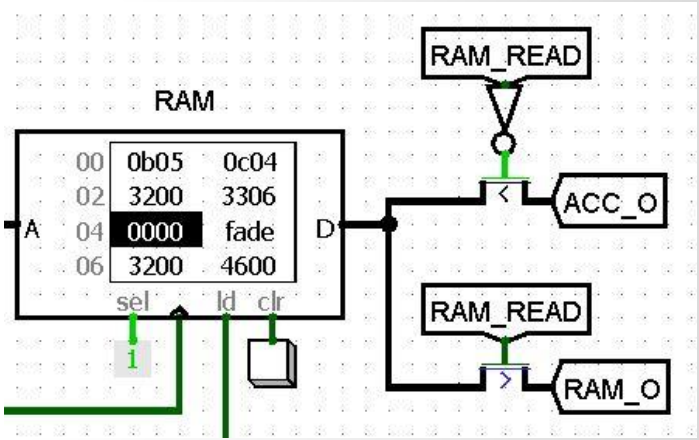
- O segundo clock indica a incrementação do PC;
- No terceiro clock, temos o armazenamento da instrução contida em 01;
- No quarto clock, temos o carregamento do endereço armazenado no acumulador no endereço 04;

endereço:	instrução:
00	0b05
01	0c04
02	3200
03	3306
04	0000
05	fade
06	3200
07	4600

IR armazena 0c04



Instrução 0x0C indica carregar dado armazenado no acumulador no endereço 0x



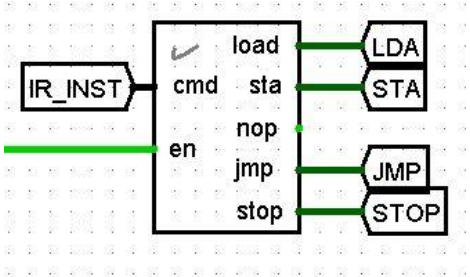
A instrução “fade” será executada no endereço 04

Sétimo clock

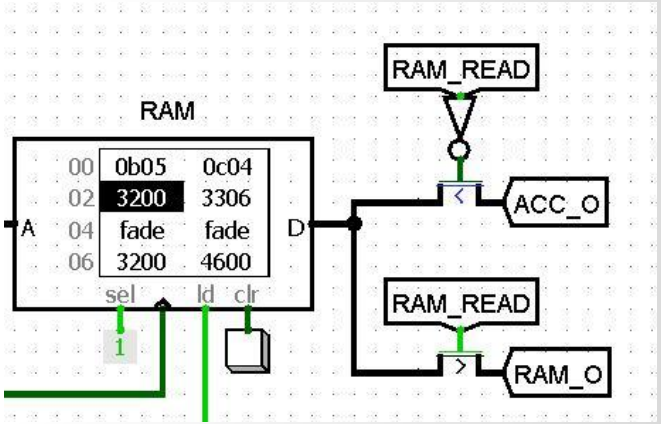
- O quinto clock indica a incrementação do PC;
- No sexto clock, temos o armazenamento da instrução contida em 02;
- No sétimo clock, temos a instrução 0x32, que nada será realizado;

endereço:	instrução:
00	0b05
01	0c04
02	3200
03	3306
04	0000
05	fade
06	3200
07	4600

IR armazena 3200



Instrução 3200 indica nenhuma operação a ser realizada



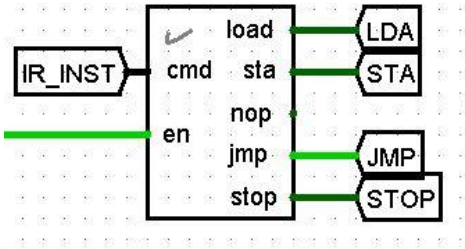
nada será realizado

Décimo clock

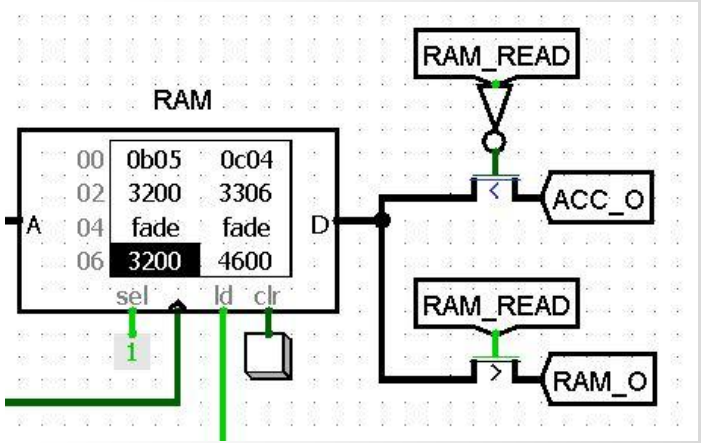
- O oitavo clock indica a incrementação do PC;
- No nono clock, temos o armazenamento da instrução contida em 03;
- No décimo clock, temos a instrução 0x33, que pulará para o endereço 06;

endereço:	instrução:
00	0b05
01	0c04
02	3200
03	3306
04	0000
05	fade
06	3200
07	4600

IR armazena 3200



Instrução 3306 indica salto não-condicional para o endereço 06



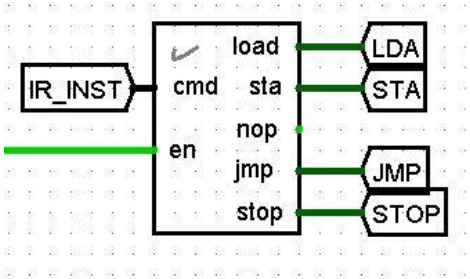
salto para o endereço 06

Décimo-terceiro clock

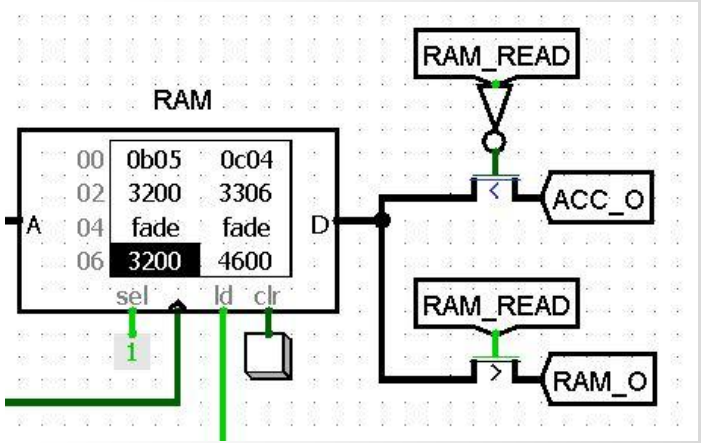
- O décimo-primeiro clock indica a incrementação do PC;
- No décimo-segundo clock, temos o armazenamento da instrução contida em 06;
- No décimo-terceiro clock, temos a instrução 0x32, que nada será realizado;

endereço:	instrução:
00	0b05
01	0c04
02	3200
03	3306
04	0000
05	fade
06	3200
07	4600

IR armazena 3200



Instrução 3200 indica nenhuma operação a ser realizada



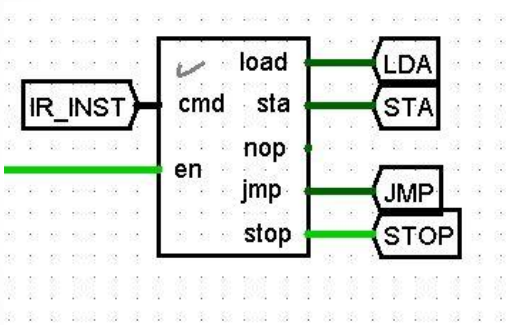
nada será realizado

Décimo-sexto clock

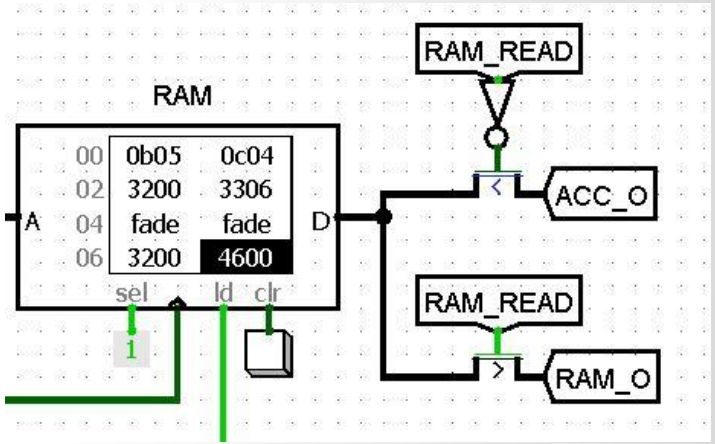
- O décimo-quarto clock indica a incrementação do PC;
- No décimo-quinto clock, temos o armazenamento da instrução contida em 07;
- No décimo-sexto clock, temos a instrução 0x46, que indica STOP;

endereço:	instrução:
00	0b05
01	0c04
02	3200
03	3306
04	0000
05	fade
06	3200
07	4600

IR armazena 3200



Instrução 4600
indica STOP



nada será realizado

STOP

Como a instrução lida foi STOP, observamos que o Demultiplexer nunca mudará de estado.
Portanto nem PC será incrementado, dando fim ao programa.