Terceiro Exercício-Programa (EP 3)

MAC329 – Álgebra Booleana e Aplicações

Marcelo S. Reis [1]
Junior Barrera [1]
Gabriela E.F. Sanada [1]

[1] Instituto de Matemática e Estatística, Universidade de São Paulo.
msreis@ime.usp.br

19 de maio de 2013

1 Introdução

Um ciclo de instrução (em inglês, *fetch-decode-execute cycle* ou *FDX*) é o ciclo básico de operação de um computador; ele é constituído das seguintes etapas:

1. busca de uma instrução na memória (*fetch*);
2. decodificação da instrução, determinando as ações exigidas pela mesma;
3. execução das ações determinadas pela instrução;
4. repetição da ação do item 1.

Toda CPU executa ciclos de instrução de forma contínua e sequencial, do momento em que o computador é inicializado até que o mesmo seja desligado. Um modelo de arquitetura que implementa o ciclo de instrução do computador HIPO foi apresentado na segunda aula desta disciplina.

1.1 Objetivos

Neste Exercício-Programa (EP) implementaremos os principais elementos do ciclo de instrução de nosso computador HIPO. Assim como na etapa anterior, a implementação do ciclo de instrução deverá ser feita em Logisim [2].

Na próxima seção descreveremos em detalhes como deverá ser a dinâmica de nosso ciclo de instrução para este EP.

2 Descrição do ciclo de instrução

Um ciclo de instrução de nosso computador HIPO será constituído das seguintes etapas:

1. leitura, da memória, do par instrução/endereço apontado por PC (*program counter*), armazenando-o no IR (*instruction register*);
2. incremento do valor do PC;
3. decodificação da instrução armazenada no IR;
4. execução da instrução decodificada;
5. se a instrução executada não for STOP, retornar para o item 1.

Como as etapas do ciclo de instrução são sequenciais, a transição entre as mesmas dar-se-á através de pulsos de um *clock*. Na figura 2 mostramos um exemplo contendo dois ciclos de instrução.

A seguir, descreveremos alguns atributos dos principais componentes de nosso ciclo de instrução.

Memória. Utilizaremos o bloco RAM do Logisim, o que reduzirá bastante o tamanho final de nosso circuito; de qualquer modo, vale a observação de que alguns tipos de memória de acesso aleatório (RAM – *Random Access Memory*) são implementados utilizando flip-flops – por exemplo, a memória RAM estática (SRAM). O bloco RAM deverá ser constituído de unidades de 16 bits, com endereçamento de 8 bits. Para cada endereço de memória, se o mesmo for instrução, o primeiro byte é o código da mesma e o segundo um endereço de memória; caso contrário, os dois bytes constituem um dado.

Acumulador e IR. Implementaremos esses registradores utilizando o bloco *Register* do Logisim; ambos deverão ter 16 bits. O acumulador armazena um dado; já o IR armazena no primeiro byte um código de instrução e no segundo um endereço de memória.

PC. Será implementado com flip-flops e terá 8 bits. O PC é um registrador especial, que armazena um endereço de memória e também possui uma entrada chamada "count up", que ao receber um pulso incrementa o seu valor em uma unidade. No início da simulação, PC deverá ter como valor padrão apontar para o endereço 0x00 da memória.

Controlador. É o "cérebro" do computador. Unidade responsável pela decodificação e execução da instrução armazenada no IR. Também é responsável pelo incremento do PC e pela definição do modo de operação (leitura/escrita) da memória e dos registradores. Será implementado utilizando circuitos combinatórios e sequenciais.

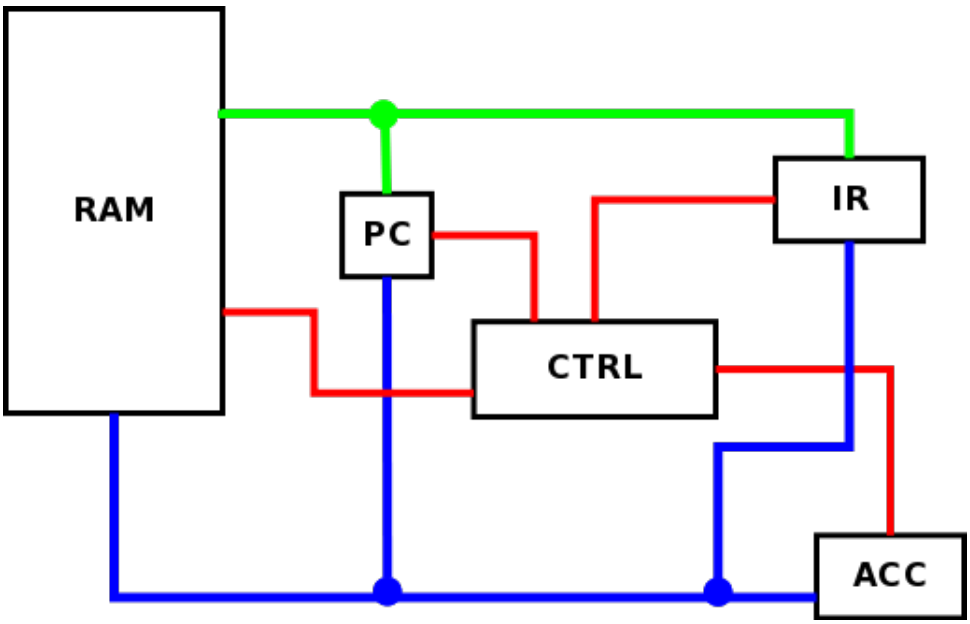

Figura 1: diagrama da arquitetura do ciclo de instrução. Em verde, o barramento de endereço, que é ligado à memória RAM, ao PC e ao IR (segundo byte deste último); em azul, o barramento de dados, que liga a memória ao PC (segundo byte do barramento), ao IR e ao acumulador (ACC); por fim, em vermelho, o barramento de controle, que liga o controlador (CTRL) a todos os demais elementos.

Barramentos. O circuito terá três tipos de barramento:

- barramento de dados (16 bits), que é ligado ao acumulador, ao IR e ao I/O do bloco RAM;
- barramento de endereço (8 bits), é ligado ao seletor de posição de memória do bloco RAM, ao IR e ao PC;
- barramento de controle, constituído de todas as ligações do controlador com os demais elementos do computador.

Na figura 1 mostramos um diagrama da arquitetura do ciclo de instrução.

2.1 Conjunto de instruções

Para esta etapa, implementaremos algumas das instruções do conjunto de instruções do HIPO [1]; as instruções que deverão ser implementadas neste EP são descritas na tabela 1. Observe que existem instruções que afetam o estado do PC, assim como instruções que mudam o estado do acumulador ou de um dado endereço de memória (vide figura 2).

Código		Instrução	Descrição
Decimal	Hexadecimal		
11	0x0B	{`LDA`} `AB`	Carrega dado do endereço `AB` no acumulador
12	0x0C	{`STA`} `AB`	Carrega acumulador no endereço `AB`
50	0x32	{`NOP`}	Sem operação
51	0x33	{`JMP`} `AB`	Salto não-condicional para o endereço `AB`
70	0x46	{`STP`}	Fim da execução

Tabela 1: Instruções HIPO que deverão funcionar neste EP.

2.2 Recomendações e dicas

- Para utilizar o bloco RAM do Logisim, basta setar o endereço desejado no seletor de posição de memória, selecionar se deseja ler ou escrever, e retirar ou colocar o dado no I/O de dados.
- Procure tomar cuidados que evitam bugs no Logisim; por exemplo, não deixar portas lógicas com entradas desconectadas, não criar muitos níveis de subcircuitos, setar pinos com apenas 2 estados (em casos em que o *don't care* não se aplicar). Algumas outras dicas que aprendemos com os alunos, para contornar algumas instabilidades do Logisim:
  - resetar o estado do circuito (CTRL+R) antes de iniciar uma simulação;
  - apagar um subcircuito e substituí-lo por um do mesmo tipo também funciona em alguns casos.
- Inicie o projeto do ciclo de instrução sem se preocupar em executar de fato as ações das instruções em si; isto é, inicialmente implemente o circuito como se todas as instruções fossem `NOP`.

3 Entrega do EP e observações

3.1 Observações importantes

- Este EP deverá ser feito em grupos de 4 pessoas, de preferência mantendo a mesma equipe do EP anterior; em casos excepcionais e com autorização do professor / monitor, pode-se aceitar grupos com menos integrantes.
- Organização do código é muito importante; para isso divida o seu circuito principal em subcircuitos que serão utilizados pelo circuito principal (e/ou por outros subcircuitos). Utilize *splitters* para diminuir a poluição visual dos barramentos e encapsule subcircuitos em contextos que exigem visualizá-los em um nível de abstração mais elevado.

- Documente adequadamente o seu circuito, através de comentários em um arquivo texto README que deverá acompanhar o código-fonte. Faça também (breves) comentários no próprio circuito, para facilitar o entendimento do mesmo.
- Recomenda-se que as dúvidas sejam discutidas no fórum da disciplina no Moodle.

3.2 Data de entrega

Este EP deverá ser entregue até o dia **9 de junho (domingo)**, através do fórum da disciplina no Moodle. Após esse dia, o sistema não permitirá a entrega deste EP.

Deverá ser entregue um zip ou tar.gz, contendo o arquivo `.circ` do código do ciclo de instrução descrito na seção 2, além de um arquivo README contendo o comentários que acharem pertinentes para o entendimento da implementação. Apenas um membro do grupo deverá subir o arquivo; portanto, lembrem-se de colocar no arquivo README o nome completo e o número USP de todos os integrantes do grupo.

Referências

[1] Valdemar W. Setzer. The HIPO computer – a tool for teaching basic computer principles through machine language. `http://www.ime.usp.br/~vwsetzer/hipo/hipo-descr.html`, 2000.

[2] Carl Burch. Logisim – a graphical tool for designing and simulating logic circuits. `http://ozark.hendrix.edu/~burch/logisim/index.html`, 2001.

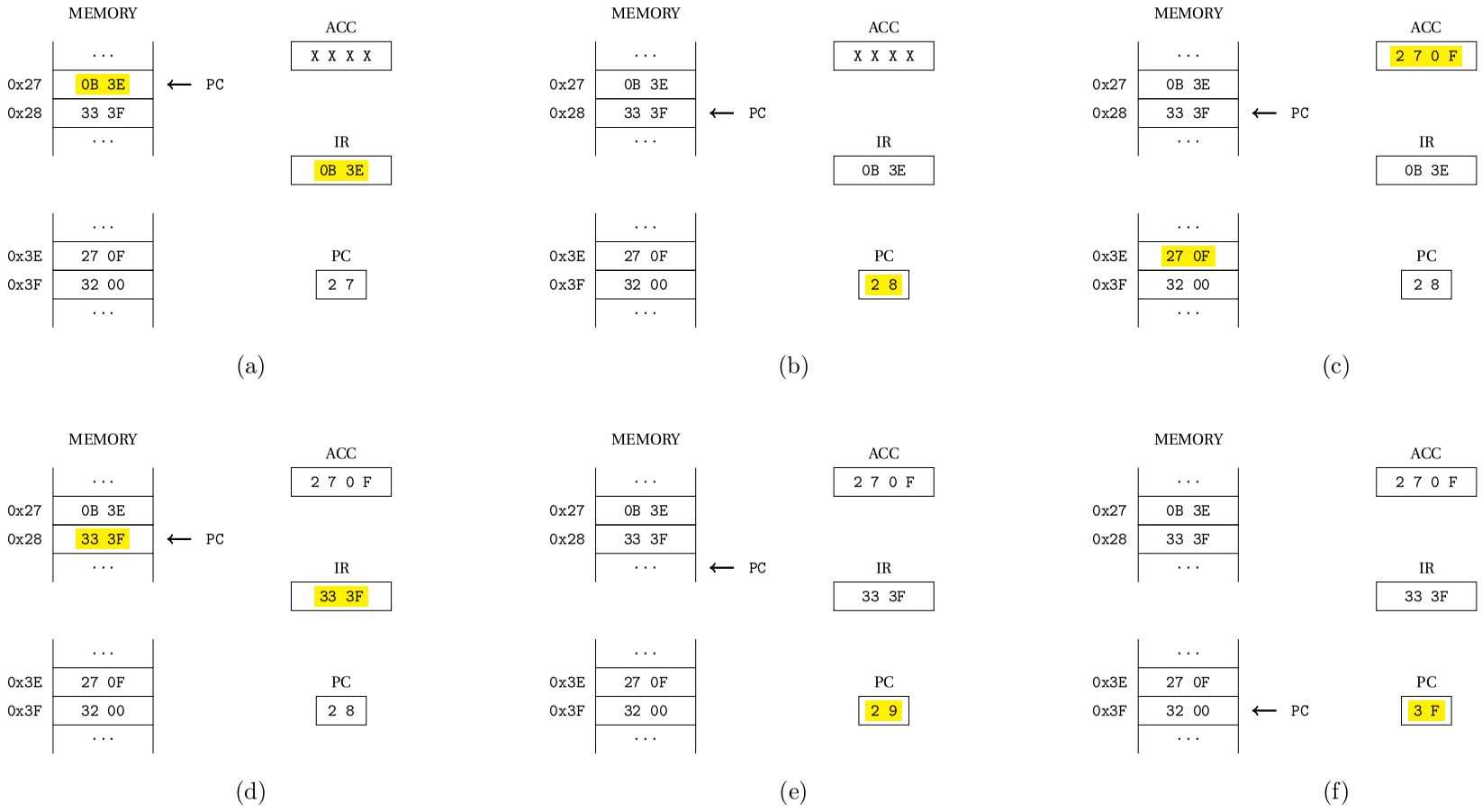

Figura 2: exemplo de dois ciclos de instrução. Figura 2(a): a instrução {`LDA`} `62` é carregada no registrador IR; Figura 2(b): o PC é incrementado para apontar para a próxima instrução; Figura 2(c): a instrução carregada em IR é decodificada e executada, ou seja, o dado da posição 62 (+9999) é carregado no acumulador; Figura 2(d): a instrução {`JMP`} `63` é carregada no registrador IR; Figura 2(e): o PC é incrementado para apontar para a próxima instrução; Figura 2(f): a instrução carregada em IR é decodificada e executada, ou seja, PC é atualizado para apontar para o endereço 63.