# Trabalho de Organização de Computadores I

14 de abril de 2014

## 1 Introdução

A empresa SciFi Games é nova no mercado de videogames. Sua equipe, composta por projetistas de hardware, está planejando o lançamento do novo videogame *KronosX*. A empresa já desenvolveu todo o projeto de arquitetura do sistema de computação, visto que sua equipe é formada por especialistas nesta área. Essa arquitetura compreende três unidades: o processador (CPU), a memória de dados e a memória de instruções.

Todavia, a SciFi Games está tendo dificuldades para sintetizar o circuito desenvolvido por sua equipe e, por meio de recomendações, resolveu contratar sua equipe de desenvolvedores de hardware para implementar a arquitetura desenvolvida por seus projetistas.

Inicialmente, a SciFi Games deseja um protótipo para testar a arquitetura antes de realizar a sintetização específica do circuito e sua comercialização. Dessa forma, a tarefa imediata de sua equipe é trabalhar na produção deste protótipo, que será desenvolvido em duas entregas, datadas na seção 3.

## 2 Arquitetura do Sistema de Computação

A figura 1, esquematiza a organização do sistema de computação, fornecendo uma visão geral. Observe que, neste sistema, cada instrução é executada em um **ciclo de clock**. Essa arquitetura é conhecida como *monociclo* ou *SingleCycle*. Este sistema de computação projetado contêm três macros: o processador, a memória de dados e a memória de instruções. Tais componentes serão descritos detalhadamente a seguir.

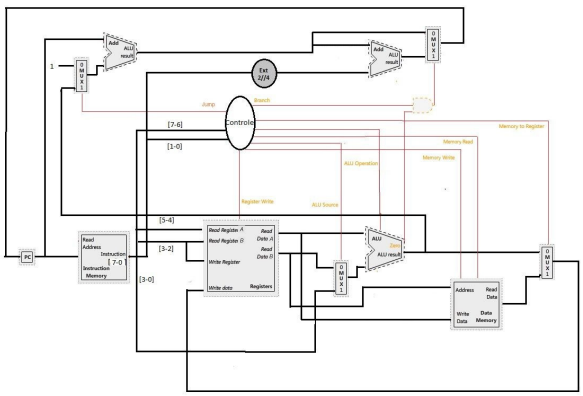


Figura 1: Visão geral do sistema de computação.

A equipe da SciFi games definiu que a arquitetura irá utilizar o formato *Little Endian*. Neste esquema,

os bits menos significativos estão nas posições mais baixas de memória. A figura 2 ilustra a ideia. É importante seguir este padrão para não criar incompatibilidades.

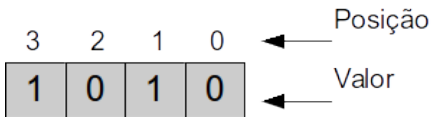


Figura 2: Escrita do valor '1010' no Formato Little Endian.

## 2.1 Memória de Dados

A memória de dados contêm 16 posições com 4 bits em cada posição. Estão inclusos, também, uma entrada para o dado (data), para o endereço de escrita (addr), para o bit de escrita (write_data), para o bit de leitura (read_data), para o clock (clk) e uma saída para o dado lido. Essas informações estão sintetizadas na tabela 1.

| Nomenclatura | Abreviação |
|---|---|
| entrada de dados | data |
| endereço de escrita | addr |
| bit de escrita | write_data |
| bit de leitura | read_data |
| clock | clk |
| saída | out_data |

Tabela 1: Nomenclatura Memória de Dados.

As operações de tanto de leitura quanto a escrita são síncronas, ou seja, dependem do clock que, em todo o circuito, **ocorre na borda de subida**. Esse esquema é mostrado na tabela 2. Uma observação importante é caso o bit de operação de leitura e escrita estiverem em '1'. Quando esta situação ocorrer, deve-se escrever o dado e, depois, ler o valor atualizado. A ilustração da memória de dados é mostrada na figura 3.

| Read_data | Clk | Out_data | Write_data | Clk | Memory |
|---|---|---|---|---|---|
| 0 | 0 | inalterada | 0 | 0 | inalterada |
| 0 | 1 | inalterada | 0 | 1 | inalterada |
| 1 | 0 | inalterada | 1 | 0 | inalterada |
| 1 | 1 | Memory[addr] | 1 | 1 | Memory[Addr]=data |

Tabela 2: Controle de sinais de entrada e saída para a memória de dados.

## 2.2 Memória de Instruções.

A arquitetura proposta pela equipe da SciFi Games consiste em ter a memória de instruções separada da memória de dados. Isso porque serão vendidos os jogos separadamente, onde estarão as instruções. Dessa forma, a memória de instruções é fixa e foi definida com tamanho de 16 posições, cada uma armazenando 8 bits. Essa memória, diferente da memória de dados, funciona de maneira assíncrona, ou seja, não há dependência do clock. Uma vez atualizado o endereço da instrução (o Program Counter (PC)), a saída é atualizada com a instrução requerida, conforme mostrado pela tabela 3. A figura 4 ilustra a memória de instruções.

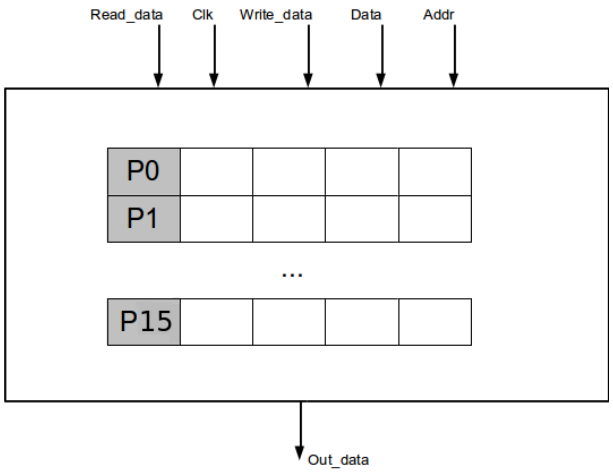


Figura 3: Ilustração da memória de dados.

| Addr | Out_data |
|---|---|
| Y | Memory[Y] |

Tabela 3: A saída da memória de instruções é atualizada com o endereço, independente do sinal de clock.

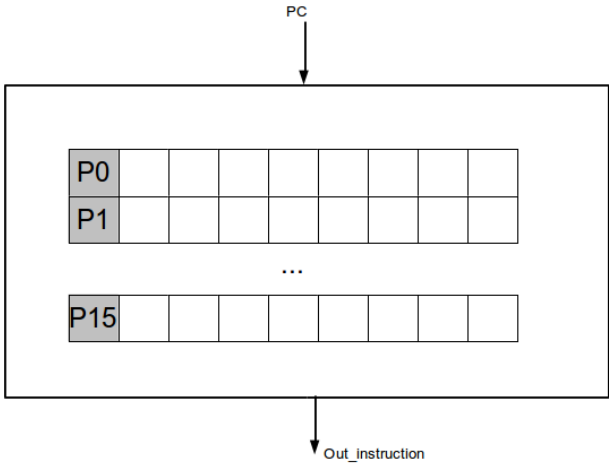


Figura 4: Ilustração da memória de instruções.

## 2.3 Processador

O processador - ou Central Processing Unit (CPU) - é a parte onde é realizada a computação em si. Essa unidade possui a ALU (Arithmetic Logic Unit), o controle e o banco de registradores. Conforme definido pela equipe de projetistas da SciFi Games, o banco de registradores contêm 4 registradores de 4 bits, sincronizados pelo clock. A especificação do controle e da ALU estão implícitas no restante desta subseção (2.3), que tratará da decodificação das instruções.

As instruções, compostas por 8 bits, são divididas em 4 tipos: instruções aritméticas, instruções de desvio incondicional, instruções de desvio condicional e instruções de operações de memória. A existência de 4 tipos de instruções requer 2 bits para representar cada instrução (InstCode) unicamente (conforme a tabela 4). Esses dois bits são os mais altos de cada instrução. Isso faz com que reste 6 bits disponíveis

para cada instrução. A seguir, será detalhado cada um desses 4 tipos.

| InstCode | Tipo da instrução |
|---|---|
| 00 | aritmética |
| 01 | desvio incondicional |
| 10 | desvio condicional |
| 11 | operação com memória |

Tabela 4: InstCode e tipos de instruções

### 2.3.1 Instruções aritméticas

Os 6 bits das instruções aritméticas são divididos em 3 partes, conforme ilustra a figura 5. O endereço de destino é o mesmo do registrador B (veja figura 1). Os 2 bits dedicados aos tipos de operações (OpCode), totalizando 4 operações, têm sua semântica mostrada na tabela 5.

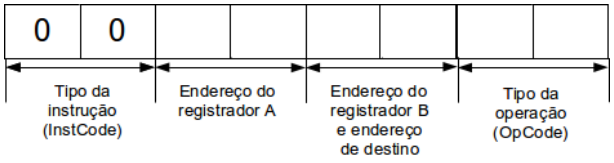


Figura 5: Ilustração das operações aritméticas.

| Opcode | Operação |
|---|---|
| 00 | adição |
| 01 | subtração |
| 10 | adição constante |
| 11 | negação |

Tabela 5: Opcode e operações aritméticas

A adição constante têm sua utilidade na inicialização de registradores. Quando a instrução for uma adição constante, os 4 últimos bits da instrução serão utilizados como constante na soma com o registrador A. O endereço de destino da soma constante permanece sendo o endereço do registrador B. A figura 6 representa a ideia.

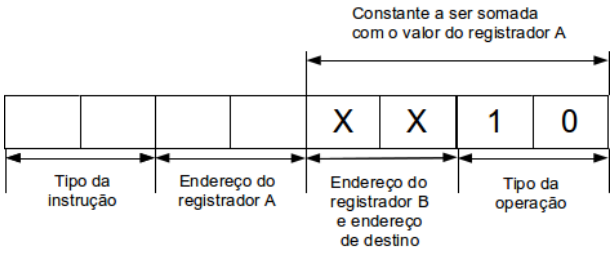


Figura 6: Ilustração da adição constante.

### 2.3.2 Instruções de Desvio Incondicional

Essa instrução permite desviar para qualquer posição na memória de instruções. Esse tipo de instrução soma os 4 bits mais baixos ao valor do registrador A e salva o resultado no PC (Program Counter). A figura 7 ilustra o formato desta instrução.

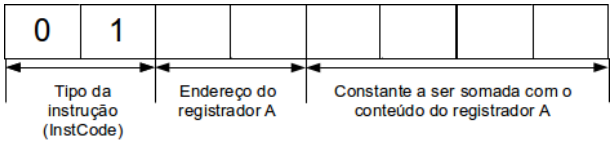


Figura 7: Ilustração do desvio incondicional.

### 2.3.3 Instruções de Desvio Condicional

Tipo de instrução que desvia o fluxo de execução caso uma condição seja satisfeita. Nesta arquitetura, a condição é que os conteúdos dos registradores A e B sejam iguais. Caso a condição seja satisfeita, o PC será incrementado os 2 bits mais baixos da instrução, conforme mostra a figura 8.

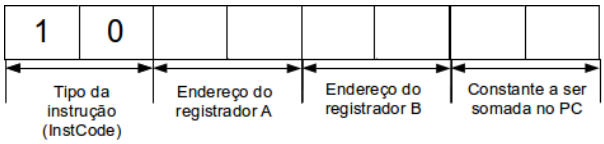


Figura 8: Ilustração do desvio condicional.

### 2.3.4 Instruções de Operações com Memória

Este tipo de instrução permite a leitura ou escrita na memória de dados. São reservados dois bits ao registrador A e dois ao registrador B. O valor armazenado no registrador A é o endereço base. O endereço base é a posição da memória de dados que sofrerá leitura ou escrita. O endereço do registrador B é o registrador de destino do valor lido, caso seja uma leitura, ou o valor a ser escrito na memória de dados, caso seja uma escrita. Os dois últimos bits representam o tipo de operação (OpCode) que pode ser consultada na tabela 6. A operação NOP representa uma operação sem efeito, ou seja, nenhuma alteração é feita no estado do sistema quando uma instrução deste tipo for executada. O esquema da instrução de operação com memória é resumido na figura 9

| Opcode | Operação |
|---|---|
| 00 | leitura da memória de dados |
| 01 | escrita na memória de dados |
| 10 | Nenhuma operação(NOP) |
| 11 | Nenhuma operação(NOP) |

Tabela 6: Opcode e operações com memória

### 2.3.5 Unidade de controle

A unidade de controle é responsável por fazer com que a semântica criada para diferenciar diversos tipos de instruções funcione. A unidade de controle recebe o tipo de instrução que está sendo executada e

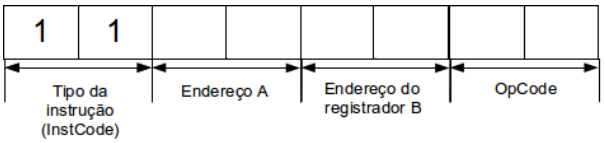


Figura 9: Ilustração de operações com memória.

fornece aos diversos componentes do sistema bits de controle, de forma a fazer com que eles reconheçam aquela instrução corretamente. Por exemplo, na figura 1, o controle chamado *ALU Source* escolhe qual valor será usado como o operando 2 da ALU: O valor que vem do registrador B ou o que está imbutido na instrução. Se for uma instrução aritmética de soma, o valor de *ALU Source* será 0, permitindo que seja repassado o valor do registrador B. Entretanto, caso for uma instrução aritmética de soma constante, seu valor será 1, escolhendo os 4 bits mais baixos da instrução como operando 2 da ALU.

Os projetistas da SciFi Games deixaram quais bits de controle serão usados, conforme a tabela 7. Entretanto, eles não projetaram o valor dos bits de controle para cada tipo de instrução, repassando essa tarefa para a sua equipe.

| Controle | Semântica | bits |
|---|---|---|
| Branch | Permite a realização do desvio condicional. | 1 |
| MemoryToRegister | Identifica o valor a ser escrito no registrador. | 1 |
| MemoryRead | Permite a leitura da memória de dados. | 1 |
| MemoryWrite | Permite a escrita na memória de dados. | 1 |
| RegisterWrite | Permite a escrita no banco de registradores. | 1 |
| ALUOperation | Identifica qual operação a ULA deve fazer. | 2 |
| Jump | Permite a realização do desvio incondicional. | 1 |
| ALUSource | Identifica qual valor será o operando B da ALU. | 1 |

Tabela 7: Bits de controle

Observe que o controle *ALUOperation* tem 2 bits. O motivo é o número de operações da ALU (4 operações), conforme discutido na seção 2.3.1 acerca do tipo de instrução aritmética. Esse sinal faz com que a ALU realize a operação pedida na instrução.

# 3 Entregas

Conforme combinado com a empresa SciFi Games, sua equipe fará a entrega em duas partes, conforme indicado na tabela 8. Será feita ainda uma demonstração com a data a ser definida.

| Entrega | Data | Valor | Resumo da entrega |
|---|---|---|---|
| I | 21/05/14 | 20 | And, Extensor, Mux 2-1, PC, Banco de Registradores, Somador 4 bits Controle, Memória de Dados, Memória de Instruções, ALU |
| II | 01/06/14 | 5 | Sistema conectado e testes |
| Demonstração | a definir | 5 | Apresentar o sistema na FPGA. |

Tabela 8: Divisão das entregas.

Os detalhes de cada entrega serão discutidos, individualmente, nas seções que seguem.

## 3.1 Entrega I

Para esta entrega, você deverá implementar componentes básicos do processador. O objetivo dessa entrega é ter os componentes prontos para a próxima entrega. Cada componente terá pontuação máxima de acordo com a tabela 9.

| Componente | Valor |
|---|---|
| And | 1 |
| Extensor | 1 |
| Mux 2-1 | 2 |
| PC | 1 |
| Banco de Registradores | 3 |
| Somador 4bits | 2 |
| Controle | 5 |
| Memória de Dados | 2 |
| Memória de Instruções | 1 |
| ALU | 2 |

Tabela 9: Valor de cada componente da entrega I.

Você deverá utilizar as interfaces (entradas e saídas) idênticas às fornecidas na figura 10. **Componentes com interfaces diferentes serão desconsiderados, ou seja, receberão nota 0**. As nomenclaturas das entradas e saídas poderão ser modificadas conforme você desejar (mudar *clk* para *clock*, por exemplo). Porém, é recomendado não alterar totalmente o nome para não se perder o significado da entrada ou saída.

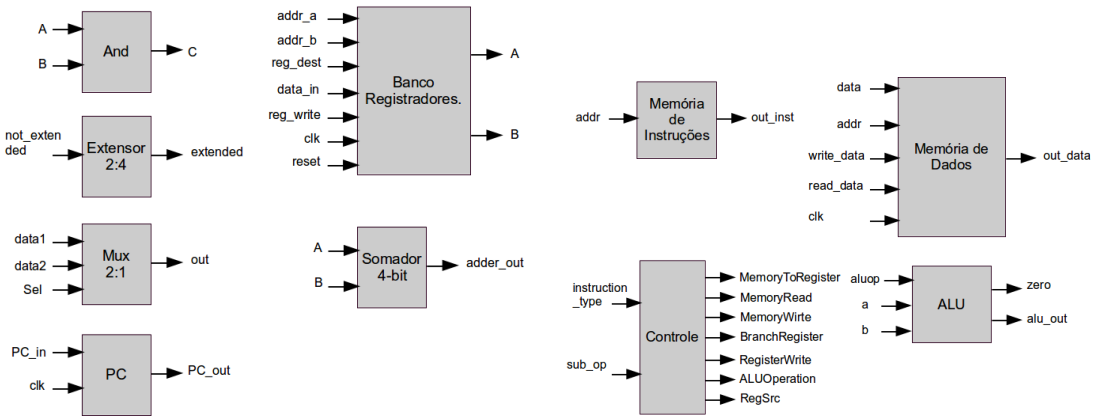


Figura 10: Interfaces dos componentes da entrega I.

## 3.2 Entrega II

Para esta entrega, você deverá integrar todas as partes desenvolvidas, a fim de produzir o protótipo pedido. A junção das partes deverá ficar semelhante a figura 1. O objetivo é o entendimento de uma arquitetura de computação, de forma a praticar e rever os conceitos abordados na disciplina. A interface para o seu sistema deverá ser idêntica a da figura 11. Você deverá entregar, junto com o sistema integrado, dois testes com, no mínimo, quatro instruções por teste. Para cada teste, você deverá descrever o que as instruções realizam, qual o resultado esperado e qual o resultado obtido na simulação.

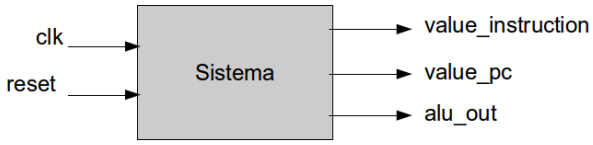


Figura 11: Interface do sistema da entrega II.

### 3.3 Demonstração

Além das documentações, sua equipe deverá apresentar, em uma entrevista, o processador executando em uma FPGA. O teste desta entrevista será disponibilizado no dia 21/05/2014 e será o mesmo para todas as equipes. A data da demonstração ainda será definida na aula do dia 02/06/2014 ou 04/06/2014. A FPGA a ser usada será a Altera Cyclone DE2 (EP2C35F672C6). A tabela 10 mostra como deverá ser mapeado as entradas e saídas do sistema, presentes na figura 11, na FPGA. Você deverá fazer uma conversão dos pinos, utilizando o arquivo da tabela de pinos disponível no 'moodle'. Por exemplo, o pino SW0 é convertido para o valor PIN_N25 na sintetização.

| Entrada | Pino FPGA |
|---|---|
| clk | SW0 |
| reset | SW1 |
| value_instruction[0] | LEDG0 |
| value_instruction[1] | LEDG1 |
| value_instruction[2] | LEDG2 |
| value_instruction[3] | LEDG3 |
| value_instruction[4] | LEDG4 |
| value_instruction[5] | LEDG5 |
| value_instruction[6] | LEDG6 |
| value_instruction[7] | LEDG7 |
| value_pc[0] | LEDR0 |
| value_pc[1] | LEDR1 |
| value_pc[2] | LEDR2 |
| value_pc[3] | LEDR3 |
| alu_out[0] | LEDR4 |
| alu_out[1] | LEDR5 |
| alu_out[2] | LEDR6 |
| alu_out[3] | LEDR7 |

Tabela 10: Mapeamento dos pinos para demonstração.

## 4 Considerações

- Comece a fazer o trabalho.
- O trabalho deverá ser feito em grupos de **no máximo 4 pessoas**.
- O trabalho deverá ser feito em VHDL ou Verilog.
- O trabalho deverá ser feito usando o software Quartus disponibilizado no moodle.
- Para todas as entregas, a equipe deverá criar uma pasta para cada componente. Dentro da pasta deverá conter o projeto do Quartus (código e testes). A figura 12 ilustra como deverá ser feito.

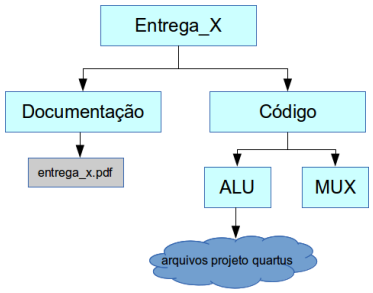


Figura 12: Organização dos diretórios.

- **Cada entrega deverá seguir o modelo disponibilizado no "moodle". Qualquer entrega que não obedeça o modelo especificado, sofrerá penalidades. Dessa forma, é fortemente recomendado seguir o modelo proposto.**
- **É expressamente proibido utilizar tipos prontos de bibliotecas como solução para os componentes pedidos.** Entretanto, é permitido usar bibliotecas para auxiliar no desenvolvimento do componente. Por exemplo, você não poderia usar o *mux* da biblioteca *lpm* de VHDL pois ele é um dos componentes a ser produzidos, mas você poderia usar o *FlipFlop-D* da mesma biblioteca para auxiliar na construção do banco de registradores.
- Não utilize bibliotecas que não sejam as padrões.
- Dúvidas? Escreva no fórum e aguarde elas serem respondidas. Sua dúvida pode ser útil a muitos.