Universidade Federal de Alagoas - UFAL

Instituto de Computação - IC

Arquitetura e Organização de computadores
Especificações de Projeto

Maceió - 2024

Organização e Arquitetura de computadores
Especificações de Projeto

Documento desenvolvido pela monitoria da disciplina Infra-Estrutura de Hardware do Cin/UFPE sob orientação da professora Drª. Edna Barros. Modificado pelo professor Erick Barboza e Bruno Nogueira.

Índice

Capítulo 1 - Especificação Inicial

Neste capítulo são apresentados os procedimentos que devem ser seguidos para a construção das duas primeiras etapas do projeto do processador descrito no segundo capítulo desta especificação. Para essas duas etapas os grupos devem usar cartolinas que, preferencialmente, terão a cor branca, ou gerar arquivos de imagem vetorial (.svg, .eps, .pdf) que serão projetados no quadro. Caso exista alguma dúvida as equipes deverão consultar o monitor.

Projetando o Processador (caminho de dados)

Nesta etapa os grupos deverão projetar na cartolina/arquivo o datapath de seu processador. O desenho apresentado deverá conter todos os circuitos que comporão o projeto e suas ligações. Além disso, o grupo deverá saber o sentido de se usar cada um dos componentes.

1.1.1 - Descrição dos módulos

Os circuitos que irão compor o projeto devem ser desenhados na cartolina/arquivo tomando como base a figura 1 do Capítulo 2 desta especificação. A quantidade e quais elementos serão desenhados irá depender da forma como cada grupo decidir implementar o conjunto de instruções que é pedido neste trabalho. Apesar dos grupos terem a liberdade de escolher quantas unidades irão compor o circuito do processador, a arquitetura desenvolvida deve dar sentido a cada um dos componentes que a forma. Além disso, é recomendável que o projeto dê ênfase ao melhor aproveitamento de todos os seus componentes, ou seja, não adianta fazer um circuito com várias unidades que façam pouca ou nenhuma atividade. Deve-se sempre pensar em economia de hardware e rapidez de execução das instruções.

1.1.2 - Ligações dos Módulos

Todas as ligações que forem relevantes ao entendimento de como cada instrução é executada deverão estar presentes no desenho da cartolina/arquivo, sendo que todos os sinais que partem ou chegam à unidade de controle deverão obrigatoriamente ser apresentados . Devem estar presentes também os sinais que interligam as demais unidades.

Observação: Os sinais que saem da unidade de controle deverão estar nomeados, pois isso facilita no entendimento geral do circuito.

1.1.3 - Sentido do Circuito

A equipe deverá estar consciente de como cada uma das instruções pedidas na especificação é executada no processador, ou seja, cada integrante deverá saber quais as unidades estão envolvidas nos diferentes estágios da instrução e como cada uma delas se comporta.

Esta etapa do projeto deverá ser apresentada ao professor durante uma aula com data definida no cronograma da disciplina. Todos os integrantes deverão estar presentes e serão questionados quanto à forma como o processador opera suas instruções.

Capítulo 2 - Especificação da CPU

2.1 - Descrição Preliminar

O projeto a ser entregue consiste na implementação de um processador, a qual poderá ser feita com base na implementação descrita no livro texto e na figura abaixo apresentada. O processador deverá incluir três blocos básicos: unidade de processamento, unidade de controle e memória. A seguir é dada uma descrição sucinta do processador a ser projetado.

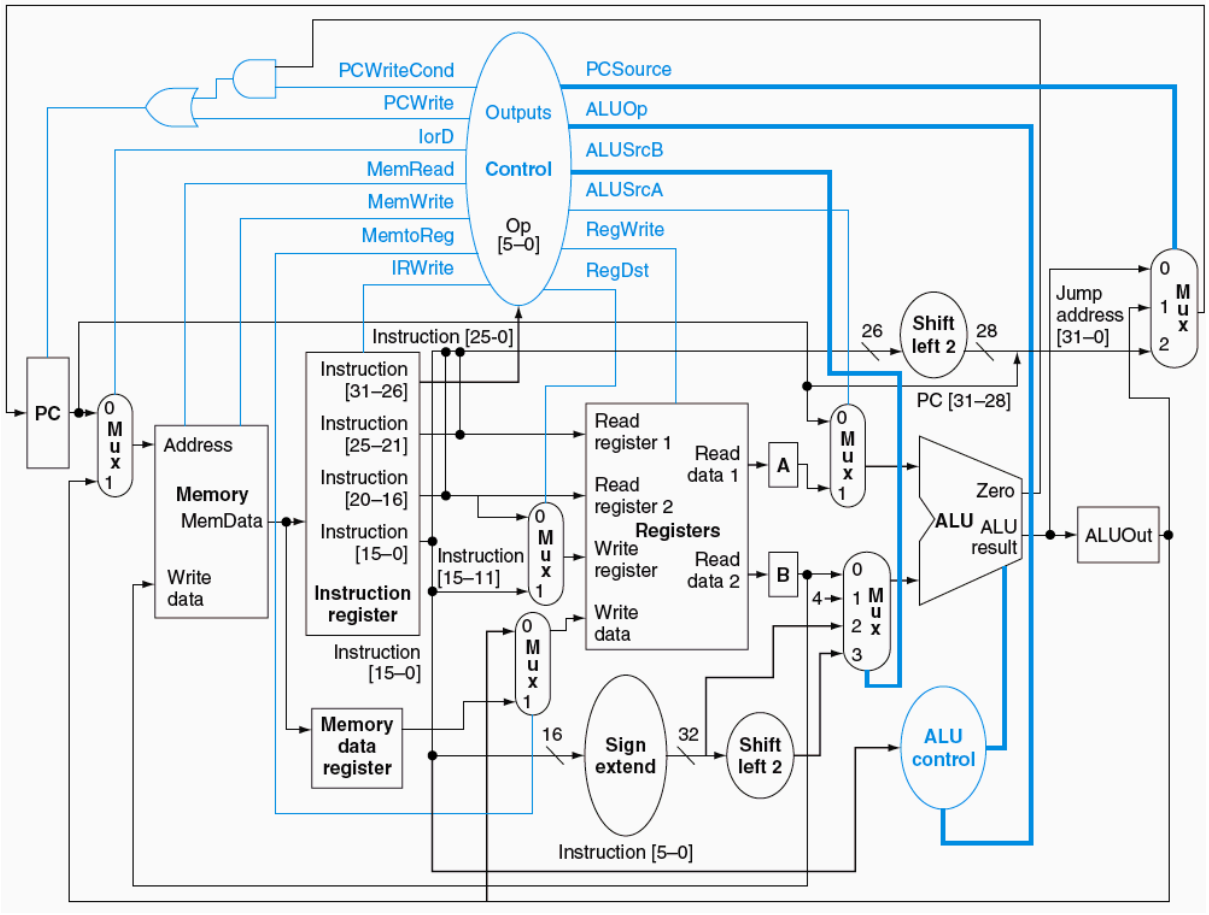

Figura 1. Circuito base para construção do projeto

Inicialmente, será projetada a unidade de processamento e a seguir a unidade de controle. A unidade de processamento será desenvolvida a partir da utilização dos seguintes componentes:

1. ALU
2. Registrador de Deslocamento
3. Registrador de Instruções
4. Banco de Registradores
5. Memória

Após o projeto da unidade de processamento com a definição de todos os seus componentes e suas respectivas portas de entrada e saída, deverá ser projetada a unidade de controle, que será implementada como uma máquina de estados finita (FSM). O processador possui 32 registradores de propósito geral, sendo, cada um, de 32 bits. Um subconjunto das instruções do MIPS deverá ser implementado, de acordo com esta

especificação. As instruções estão descritas neste capítulo e agrupadas por formato. Um resumo dos formatos existentes pode ser visto na Figura 2.

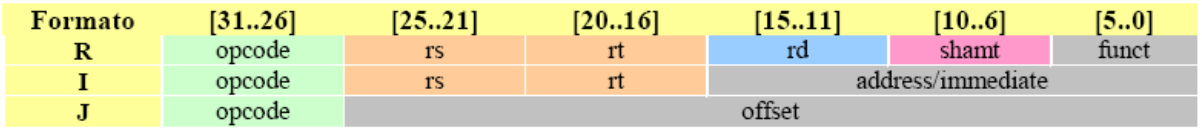

Figura 2. Formatos de Instruções.

Tabela 1. Instruções Formato R

Assembly	Opcode	rs	rt	rd	shamt	funct	Comportamento
`add rd, rs, rt`	0x0	rs	rt	rd	0x0	0x20	Rd ← rs + rt
`and rd, rs, rt`	0x0	rs	rt	rd	0x0	0x24	Rd ← rs & rt
`div rs, rt`	0x0	rs	rt	-	0x0	0x1a	**(Ver divisão)**
`mult rs,rt`	0x0	rs	rt	-	0x0	0x18	**(Ver multiplicação)**
`jr rs`	0x0	rs	-	-	0x0	0x8	PC ← rs
`mfhi rd`	0x0	-	-	rd	0x0	0x10	rd ← hi
`mflo rd`	0x0	-	-	rd	0x0	0x12	rd ← lo
`sll rd, rt, shamt`	0x0	-	rt	rd	shamt	0x0	rd ← rt << shamt
`sllv rd, rs, rt`	0x0	rs	rt	rd	0x0	0x4	rd ← rs << rt
`slt rd, rs, rt`	0x0	rs	rt	rd	0x0	0x2a	rd ← (rs < rt) ?1 :0
`srl rd, rt, shamt`	0x0	-	rt	rd	shamt	0x2	rd ← rt >> shamt
`sub rd, rs, rt`	0x0	rs	rt	rd	0x0	0x22	rd ← rs - rt
`Rte`	0x0	-	-	-	0x0	0x13	PC EPC
`Push`	0x0	-	rt	-	0x0	0x5	**(Ver push e pop)**
`Pop`	0x0	-	rt	-	0x0	0x6	**(Ver push e pop)**

Tabela 2. Instruções Formato I

Assembly	Opcode	rs	rt	End/Imediato	Comportamento
`addi rt, rs, imediato`	0x8	rs	rt	imediato	Rt ← rs + imediato**
`beq rs,rt, offset`	0x4	rs	rt	offset	Desvia se rs == rt (ver desvios)
`bne rs,rt, offset`	0x5	rs	rt	offset	Desvia se rs != rt (ver desvios)
`ble rs,rt,offset`	0x6	rs	rt	offset	Desvia se rs <= rt (ver desvios)
`bgt rs,rt,offset`	0x7	rs	rt	offset	Desvia se rs > rt (ver desvios)
`lb rt, offset(rs)`	0x20	rs	rt	offset	Rt ← byte Mem[offset + rs]
`lh rt, offset(rs)`	0x21	rs	rt	offset	Rt ← halfword Mem[offset + rs]
`lui rt, imediato`	0xf	-	rt	imediato	Rt ← imediato << 16
`lw rt, offset(rs)`	0x23	rs	rt	end	Rt ← Mem[offset + rs]
`sb rt, offset(rs)`	0x28	rs	rt	offset	Mem[offset +rs] byte[rt] (ver stores)
`slti rt, rs, imediato`	0xa	rs	rt	imediato	Rt ← (rs < imediato) ?1 :0
`sw rt, offset(rs)`	0x2b	rs	rt	offset	Mem[offset +rs] ← rt

***** o valor de 'imediato' deve ser estendido para 32 bits, estendendo seu sinal (bit mais significativo da constante).***

Tabela 3. Instruções Formato J

Assembly	Opcode	Offset	Comportamento
`j offset`	0x2	offset	(ver desvios)
`jal offset`	0x3	offset	(ver desvios)

2.2 - A Pilha

Para permitir a chamada de rotinas reentrantes e recursivas, será possível o armazenamento do registrador 31 numa estrutura do tipo pilha a qual será implementada numa parte da memória como mostrado na figura abaixo:

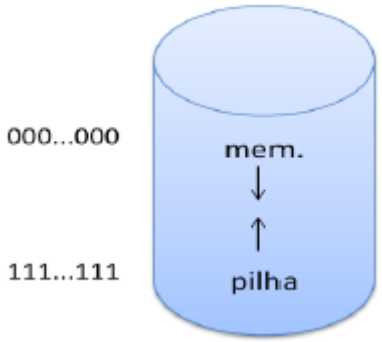

Figura 3. Pilha

O endereço do topo da pilha será guardado no registrador 29 (vinte e nove) do banco de registradores, que funciona como o SP (Stack Pointer). Quando o reset é ativado este registrador deverá apontar para o endereço onde começa a pilha (nesta versão: 227). O salvamento e recuperação do registrador 31 na pilha é sempre feito por software (código assembly).

2.3 - Divisão

A divisão deve ser implementada como um componente à parte, o **Divisor**, não sendo recomendada a reutilização da ALU principal do processador para efetuar qualquer operação referente a esta função. Este componente deve realizar corretamente a divisão de dois números de 32 bits, com sinal, resultando em um número de 32 bits, também com sinal.

Instrução	Descrição
div rs, rt	Realiza a divisão entre rs e rt, armazenando internamente (em dois registradores próprios do divisor) o resultado.
mfhi rd	Copia a parte alta do resultado da divisão - o resto (32 bits mais significativos) para o registrador rd.
mflo rd	Copia a parte baixa do resultado da divisão - quociente (32 bits menos significativos) para o registrador rd.

Importante: Note que o resultado da divisão inteira é armazenado em dois registradores. O registrador Hi deve conter o resto e o registrador Lo o quociente.

2.4 - Multiplicação

A multiplicação deve ser implementada como um componente à parte, o **Multiplicador**, não sendo recomendada a reutilização da ALU principal do processador para efetuar qualquer operação referente a esta função. Este componente realizará corretamente a multiplicação de dois números de 32 bits, com sinal, resultando em um número de 64 bits, também com sinal.

Instrução	Descrição
mult rs, rt	Realiza a multiplicação entre rs e rt, armazenando internamente (em dois registradores próprios do multiplicador) o resultado.
mfhi rd	Copia a parte alta do resultado da multiplicação (32 bits mais significativos) para o registrador rd.
mflo rd	Copia a parte baixa do resultado da multiplicação (32 bits menos significativos) para o registrador rd.

Importante: Note que o resultado da multiplicação inteira é armazenado em dois registradores. O registrador Hi deve conter os 32 bits mais significativos e o registrador Lo os bits menos significativos.

2.5 - Desvios

Os desvios do MIPS podem ser classificados como desvios condicionais ou incondicionais. Os desvios condicionais realizam algum teste ou verificação para, somente então, executar o desvio. Os incondicionais sempre executam o desvio. O cálculo do endereço de destino é realizado diferentemente nos dois tipos de desvio:

Desvios condicionais: O valor de 'offset' é multiplicado por 4 e somado ao PC + 4. O resultado desta operação é o endereço de destino do salto (desvio), caso o resultado do teste condicional seja verdadeiro.

*Exemplo: beq r3, r4, 0xf - Supondo que r3 e r4 sejam, ambos, iguais a 0x3a e que o PC + 4 seja 0x1c, a próxima instrução a ser executada é a que ocupa a posição (0xf * 4) + 0x1c, que é a 0x58.*

Desvios incondicionais: No caso de instruções que utilizem offset, este valor deve ser multiplicado por 4, resultado em um endereço de 28 bits. Como todo endereço deve possuir 32 bits, os 4 bits restantes vêm do PC atual, representando os 4 bits mais significativos do endereço de destino. No caso de instruções que utilizam registrador, o conteúdo do registrador já é o próprio endereço de destino.

*Exemplos: jal 0x1a2b - Supondo que o PC atual seja igual a 0xba12fa00, o endereço de destino será igual a [(PC & 0xf0000000) | (0x1a2b * 0x4)], que é igual a 0xb00068AC. Note que os 4 bits mais significativos do PC (1011) foram conservados. jr 6 - Supondo que o r6 seja igual a 0x1a2b, o endereço de destino será igual ao próprio valor de r6, neste caso, 0x1a2b.*

Instruções		
j	offset	Desvia, incondicionalmente, para o endereço calculado.
jal	offset	Desvia para endereço calculado, porém salva o endereço da instrução subsequente ao jal (endereço de retorno) no registrador r31 (SEMPRE).
jr	rs	Desvio incondicional para o endereço apontado por rs.

2.6 - Stores

Nas instruções sb (store byte) e sh (store halfword) deve-se ter cuidado para não sobrescrever a palavra toda (4 bytes) na memória. A memória tem como entradas um endereço de 4 bytes e uma palavra para ser armazenada. Deve-se apenas salvar um byte (no caso de um sb) no endereço especificado e não substituir a palavra toda que está armazenada na memória pelo byte seguido de zeros.

2.7 Push e Pop

As instruções push e pop servem respectivamente para armazenar registradores na pilha da memória e restaurar valores de registradores que estão na pilha. A utilização destas instruções implica em atualizar também o valor do registrador SP (registrador 29). A pilha cresce de endereços maiores para menores como explicado anteriormente. Portanto além de armazenar (ou ler) um valor da memória, o registrador SP deve ser atualizado.

Exemplo: push rt – Supondo que o registrador SP aponte para o endereço 227(decimal), ele deve ser decrementado em 4 (223) e o conteúdo de rt deve ser armazenado nesta posição de memória(223).

Exemplo: pop rt – Supondo que o registrador SP aponte para o endereço 223(decimal),o conteúdo desta posição de memória deve ser armazenado no registrador rt e o registrador SP deve ser incrementado em 4 (227) .

2.8 - Componentes Extras a serem considerados

Além dos componentes descritos em sala e apresentados na Figura 1, os grupos poderão considerar ainda um registrador de deslocamento.

2.8.1 - Registrador de Deslocamento

O registrador de deslocamento deve ser capaz de deslocar um número inteiro de 32 bits para esquerda e para a direita. No deslocamento a direita o sinal pode ser preservado ou não. O número de deslocamentos pode variar entre 0 e 31 e é especificado na entrada n (5 bits) do registrador de deslocamento. A funcionalidade desejada do registrador de deslocamento é especificada na entrada shift (3 bits menos significativos) conforme figura abaixo.

shift	Descrição
000	Nada a fazer.
001	Load no registrador.
010	Shift a esquerda **n** vezes.
011	Shift a direita lógico **n** vezes.
100	Shift a direita aritimético **n** vezes.
101	Rotação a direita **n** vezes.
110	Rotação a esquerda **n** vezes.

Figura 7. Registrador de Deslocamento

2.8.2 - Unidade Lógica e Aritimética (ALU)

A Unidade Lógica e Aritmética (ALU) é um circuito combinacional que permite a operação com números de 32 bits na notação complemento a dois. A funcionalidade é especificada pela entrada F conforme descrito na tabela abaixo.

Função	Operação	Descrição	Flags
000	S = A	Carrega A	Z, N
001	S = A + B	Soma	Z, N, O
010	S = A – B	Subtração	Z, N, O
011	S = A and B	And lógico	Z
100	S = A + 1	Incremento de A	Z, N, O
101	S = not A	Negação de A	Z
110	S = A xor B	OU exclusivo	Z
111	S = A comp B	Comparação	EG, GT, LT

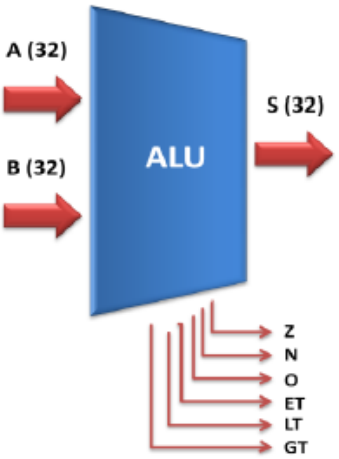

Figura 8. ALU