Escalonamento Dinâmico de Instruções

O objetivo deste projeto é implementar um simulador superescalar, que utiliza o algoritmo de Tomasulo com finalização fora de ordem.

Configuração do Pipeline

O pipeline possui os seguintes estágios: *Instruction Fetch*, *Issue*, *Execute* e *Writeback*. A implementação deverá considerar as seguintes unidades funcionais:

1. Somador: realiza as operações de soma e subtração, além das operações lógicas;
2. Multiplicador: realiza as multiplicações;
3. Divisor: realiza as divisões;
4. Unidade de carga (load);
5. Unidade de escrita (store);
6. 32 registradores de propósito geral, de 32 bits e com infinitas portas para cada banco de registradores.

A configuração do hardware deve ser completamente parametrizável, ou seja, o usuário poderá escolher: quantidade de cada unidade funcional, ciclos para cada instrução, etc. Todos os componentes da arquitetura devem ser parametrizados.

Instruções

- Transferências de Dados

ld	Rb, Ra	--- Rb ← [Ra]
st	Rb, Ra	--- [Ra] ← Rb
li	Ra, imediato	--- Ra← imediato
move	Ra, Rb	--- Ra ← Rb

- Fluxo de controle

beq	Rs, Rt, label	--- if Rs == Rt then PC←label
beqz	Rs, label	--- if Rs == 0 then PC←label
bne	Rs, Rt, label	--- if Rs <> Rt then PC←label
bnez	Rs, label	--- if Rs <> 0 then PC←label
bgt	Rs, Rt, label	--- if Rs > Rt then PC←label
bge	Rs, Rt, label	--- if Rs >= Rt then PC←label
bgtz	Rs, label	--- if Rs > 0 then PC←label
blt	Rs, Rt, label	--- if Rs < Rt then PC←label
ble	Rs, Rt, label	--- if Rs <= Rt then PC←label
blez	Rs, label	--- if Rs <= 0 then PC←label
b	label	--- PC ←label

- Instruções aritméticas

add	Rd, Rs, Rt	--- Rd ← Rs + Rt
addi	Rt, Rs, imediato	--- Rt ← Rs + imediato
sub	Rd, Rs, Rt	--- Rd ← Rs - Rt

subi	Rt, Rs, imediato	--- Rt ← Rs - imediato
mult	Rd, Rs, Rt	--- Rd ← Rs * Rt
multi	Rt, Rs, imediato	--- Rt ← Rs * imediato
div	Rd, Rs, Rt	--- Rd ← Rs / Rt
divi	Rt, Rs, imediato	--- Rt ← Rs / imediato

- Instruções Lógicas

and	Rd, Rs, Rt	--- Rd ← Rs AND Rt	(E lógico)
andi	Rt, Rs, imediato	--- Rt ← Rs AND imediato	(E Lógico)
or	Rd, Rs, Rt	--- Rd ← Rs OR Rt	(OU lógico)
ori	Rt, Rs, imediato	--- Rt ← Rs OR imediato	(OU Lógico)
neg	Rd, Rs	--- Rd ← - Rs	
not	Rd, Rs	--- Rd ← NOT(Rs)	(Negação bit a bit)
sll	Rd, Rs, Rt	--- Rd ← Rs << Rt	(Deslocamento a esquerda)
slr	Rd, Rs, Rt	--- Rd ← Rs >> Rt	(Deslocamento a direita)

Formato das Instruções

- Formato 1

Opcode	R	R	R	
6 bits	5 bits	5 bits	5 bits	11 bits (ou 16 bits) – não utilizados

Tipos de Instruções

OPCODE R R
OPCODE R R R

- Formato 2

Opcode	R	I ou L
6 bits	5 bits	21 bits

Tipos de Instruções

OPCODE R I
OPCODE R L

- Formato 3

Opcode	R	R	I ou L
6 bits	5 bits	5 bits	16 bits

Tipos de Instruções

OPCODE R R I
OPCODE R R L

- Formato 4

Opcode L

6 bits	26 bits

Tipo de Instrução

OPCODE L

Opcodes

ld --- 000000
st --- 000001
move --- 000010
neg --- 000011
not --- 000100
add --- 000101
sub --- 000110
mult --- 000111
div --- 001000
and --- 001001
or --- 001010
sll --- 001011
slr --- 001100

li --- 010000
beqz --- 010001
bnez --- 010010
bgtz --- 010011
blez --- 010100

addi --- 100000
subi --- 100001
multi --- 100010
divi --- 100011
andi --- 100100
ori --- 100101
beq --- 100110
bne --- 100111
bgt --- 101000
bge --- 101001
blt --- 101010
ble --- 101011

b --- 110000

Registradores

O simulador possui 32 registradores de propósito geral, além dos registradores de controle necessários. Os registradores de propósito geral possuem o seguinte formato:

R0 --- 00000
R1 --- 00001

R2 --- 00010
...
R31 --- 11111

Organização da Memória

A memória está dividida em duas regiões: dados e texto. A região de texto inicia no endereço 0, enquanto a região de dados inicia no último endereço de memória. A memória é organizada em bytes.

Entrada

A entrada do simulador é um arquivo texto contendo a parametrização da arquitetura e o código a ser executado. O formato do arquivo de entrada é o seguinte:

ARQUITETURA
    somador = *N*
    multiplicador = *N*
    divisor = *N*
    busca_de_instrucoes = *N*
    buffer_de_carga = *N*
    buffer_de_escrita = *N*

CICLOS
    add = *N*
    addi = *N*
    ...

TEXTO
    .data
        <dados>
    .text
        <código>

Saída

A saída deve conter mensagens sobre o status da arquitetura em cada ciclo de execução.

Simulador

O simulador deverá conter no mínimo os seguintes módulos:

1. Memória
2. Processador, contendo os seguintes módulos:
    1. Banco de registradores; e
    2. Pipeline.
3. Barramento de dados;
4. Barramento de controle; e
5. Clock

O clock será responsável pelo envio de um pulso aos componentes da arquitetura. Por sua vez, um componente somente entra em funcionamento quando recebe um pulso de clock.

Além disto seu simulador deverá ter um módulo que possui as seguintes funcionalidades:

1. Lê o arquivo de entrada;
2. Parametriza o simulador; e
3. Converte o programa em binário e o armazena na memória.

Simulação

Para a simulação deverá ser implementados os seguintes programas:

1. Ordenação de um vetor de 100 elementos, utilizando o método da bolha;
2. Busca binária de um elemento em um vetor de 100 elementos;
3. Multiplicação de matrizes;
4. Um código que exemplifique a finalização fora de ordem.

No dia da apresentação do trabalho serão fornecidos, pelo professor, vários programas testes, para os quais deverão ser gerados relatórios da simulação.

Implementação

A implementação deve ser impreterivelmente em C, para um sistema operacional padrão Linux. Além disto, um módulo do simulador deve ser implementado em Assembly.

Entrega

Deverá ser enviado por e-mail um pacote contendo: código fonte, manual de uso do simulador, programas testes e relatório de saída de cada programa.

NOTAS

- Qualquer tipo de cópias anularão os trabalhos completamente.
- A entrega do trabalho é impreterivelmente até as 23:59 da data marcada (veja minha página). Trabalhos entregues após a data marcada, não serão considerados.
- Códigos que não compilam, não serão considerados. Ou seja, a tais trabalhos será atribuído a nota ZERO.