dv-cpu-rv: CPU design of RISC-V

Devin

balddevin@outlook.com

2023/7/15

Content

1 Hardware Design 错误!未定义书签。

1.1 Basic Single Cycle Implementation 3

1.1.1 Deisng Diagram 3

1.2 Basic Pilelined Implementation 4

1.2.1 Design Diagram 4

1.2.2 Data Hazard: Forwarding or Bypass 4

1.2.3 Control Hazard: Branch Prediction 4

2 Appendices 4

2.1 Appendix 1: Support of Instruction Set 4

2.3 Appendix 2: Examples for Run 7

2.3.1 Example 1: Add and Store. 7

2.3.2 Example 2: Sum Less Than. 7

# Preface

The design of this CPU mainly refers to *Computer Organization and Design: The Hardware / Software Interface*: RISC-V Edition, David A.Patterson, John L. Hennessy.

The source code is also distributed to Github: [devindang/dv-cpu-rv](https://github.com/devindang/dv-cpu-rv).

# Hardware Design

## Basic Single Cycle Implementation

### Deisng Diagram



Figure 1.1 The Basic Single Cycle Implementation of CPU

The figure above shows the implementation of the basic single cycle cpu, in which the hinted lines are signals of control path, while the others are signals of data path.

## Basic Pilelined Implementation

### Design Diagram

### Data Hazard: Forwarding or Bypass

Data hazards are obstacles to pipelined execution.

### Control Hazard: Branch Prediction

# Appendices

## Appendix 1: Support of Instruction Set

The regularity of opcode:

The example of RISC-V pseudoinstructions can be found in *risc-v specification* v2.2 p109, from which we can deal with the assembly codes.

**RV32I Base Instruction Set**

Total: 47

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Type** | **Order** | **Instruction** | **Description** | **Compatibility** |
| R-type | 1 | ADD | Add. |  |
| 2 | SUB | Subtract. |  |
| 3 | SLL | Shift Left Logical. |  |
| 4 | SLT | Set Less Than. |  |
| 5 | SLTU | Set Less Than Unsigned. |  |
| 6 | XOR | Exclusive or. |  |
| 7 | SRL | Shift Right Logical. |  |
| 8 | SRA | Shift Right Arithmetic. |  |
| 9 | OR | Or. |  |
| 10 | AND | And. |  |
| I-type | 11 | JALR | Jump And Link Register. |  |
| 12 | LB | Load Byte. |  |
| 13 | LH | Load Halfword. |  |
| 14 | LW | Load Word. |  |
| 15 | LBU | Load Byte Unsigned. |  |
| 16 | LHU | Load Halfword Unsigned. |  |
| 17 | ADDI | Add Immediate. |  |
| 18 | SLTI | Set Less Than Immediate. |  |
| 19 | SLTIU | Set Less Than Immediate Unsigned. |  |
| 20 | XORI | Exclusive Or Immediate. |  |
| 21 | ORI | Or Immediate. |  |
| 22 | ANDI | And Immediate. |  |
| 23 | SLLI | Shift Left Logic Immediate. |  |
| 24 | SRLI | Shift Right Logic Immediate. |  |
| 25 | SRAI | Shift Right Arithmeic Immediate. |  |
| S-type | 26 | SB | Store Byte. |  |
| 27 | SH | Store Halfword. |  |
| 28 | SW | Store Word. |  |
| B-type | 29 | BEQ | Branch if Equal. |  |
| 30 | BNE | Branch Not Equal. |  |
| 31 | BLT | Branch Less Than. |  |
| 32 | BGE | Branch Greater or Equal. |  |
| 33 | BLTU | Branch Less Than Unsigned. |  |
| 34 | BGEU | Branch Greater or Equal Unsigned. |  |
| U-type | 35 | LUI | Load Upper Immediate. |  |
| 36 | AUIPC | Add Upper Immediate to PC. |  |
| J-type | 37 | JAL | Jump And Link. |  |
| other | 38 | FENCE |  | NO |
| 39 | FENCE.I |  | NO |
| 40 | ECALL |  | NO |
| 41 | EBREAK |  | NO |
| 42 | CSRRW |  | NO |
| 43 | CSRRS |  | NO |
| 44 | CSRRC |  | NO |
| 45 | CSRRWI |  | NO |
| 46 | CSRRSI |  | NO |
| 47 | CSRRCI |  | NO |

**RV64I Base Instruction Set (in addition to RV32I)**

Total: 15

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Type** | **Order** | **Instruction** | **Description** | **Compatibility** |
| R-type | 1 | ADDW | Add Word. |  |
| 2 | SUBW | Subtract Word. |  |
| 3 | SLLW | Shift Left Logical Word. |  |
| 4 | SRLW | Set Less Than Word. |  |
| 5 | SRAW | Set Less Than Unsigned Word. |  |
| I-type | 6 | LWU | Load Word Unsigned. |  |
| 7 | LD | Load Doubleword. |  |
| 8 | SLLI | Shift Left Logic Immediate. |  |
| 9 | SRLI | Shift Right Logic Immediate. |  |
| 10 | SRAI | Shift Right Arithmetic Immediate. |  |
| 11 | ADDIW | Add Immediate Word. |  |
| 12 | SLLIW | Shift Left Logic Immediate Word. |  |
| 13 | SRLIW | Shift Right Logic Immediate Word. |  |
| 14 | SRAIW | Shift Right Arithmetic Immediate Word. |  |
| S-type | 15 | SD | Store Doubleword. |  |

## Appendix 2: Examples for Run

### Example 1: Add and Store.

**Date**: 2023/7/23

**Hash**: a7b05c264b7f45e27a81ddc02184c6dcee29fdf9

**Description**: Given two numbers, add them and store into memory.

C code

#include "stdio.h"

int main() {

int a = 14;

int b = 15;

int c;

c = a + b;

return 0;

}

Assembly code

addi x2 x0 14; //0// 00000000111000000000000100010011

addi x3 x0 15; //1// 00000000111100000000000110010011

add x1 x2 x3; //2// 00000000001100010000000010110011

sd x1 16(x2); //3// 00000000000100010011010000100011

### Example 2: Sum Less Than.

**Date**: 2023/7/29

**Hash**: 1d28ab2a485737b8bd90fa777fd550d5183b705c

**Description**: Given a non-zero natural number N, calculate the sum of natural numbers less than N.

C code

#include "stdio.h"

int main() {

int N = 10;

int sum = 0;

for(int i=1; i<N; i++){

sum = sum+i;

}

return 0;

}

Assembly code

addi x1 x0 10; //0// 00000000101000000000000010010011

addi x2 x0 1; //4// 00000000000100000000000100010011

addi x3 x0 0; //8// 00000000000000000000000110010011

add x3 x2 x3; //12// 00000000001100010000000110110011

addi x2 x2 1; //16// 00000000000100010000000100010011

blt x2 x1 A12; //20// 11111110000100010100110011100011

sd x3 16(x1); //24// 00000000001100001011010000100011