|  |
| --- |
| rochester institute of technology |
| The Smartest Little Computer |
| Hardware and Control Unit Document |
|  |
| **Jenny Zhen & Grant Kurtz** |
| **4/28/2014** |

|  |
| --- |
| The hardware and control unit document describes the basic Register Transfer Language (RTL) for all instructions in the CPU. The control unit describes the control flow of the RTL for managing the overall operation of the CPU. |

Table of Contents

[Introduction 2](#_Toc388391156)

[Architecture 2](#_Toc388391157)

[Diagram 2](#_Toc388391158)

[Branch Prediction Table 2](#_Toc388391159)

[Construction 2](#_Toc388391160)

[Operation 3](#_Toc388391161)

[Register Transfer Language 3](#_Toc388391162)

[Fetch Phase 4](#_Toc388391163)

# Introduction

The purpose of this document is to specify the entirety of the simulated computer to test and demonstrate the effectiveness of branch prediction on a pipelined RISC architecture. The instruction set is kept simple and small but with a large enough variability to create a complete testing environment. Instructions are limited to only a single byte in width so as to keep the pipeline itself simple, to minimize the complexity necessary to add branch prediction to the pipeline.

# Architecture

## Diagram

See attached diagram at end of document.

## Branch Prediction Table

### Construction

The branch prediction table (BPT) is responsible for making accurate (albeit slow) predictions for whether an I- or J-Type instruction will branch. If it predicts taken, then the BPT will output the predicted PC to be used instead of the other generated values. The BPT is composed of 32 registers, 16 used for identification of branch statements, and 16 used for storing the new PC. Both sets of registers also contain information about historical pattern data used by the control unit to make a prediction. The algorithm for detecting jump or not jump is a Two-Level Adaptive Predictor.

The first bank of registers, known as the identification bank (I-bank), has the following format.

|  |  |  |  |
| --- | --- | --- | --- |
| Historical Values, 11-10 | 00, A-Predictor, 9-8 | 01, B-Predictor, 7-6 | PC, 5-0 |

The second bank of registers, known as the resolution bank (R-bank), has the following format.

|  |  |  |
| --- | --- | --- |
| 10, C-Predictor, 11-10 | 11, D-Predictor, 9-8 | New Program Counter, 7-0 |

### Operation

#### Querying for Existing Entries

All entries in the BPT are enumerated until an exact matching PC is found. If no such match is found, then the BPT does not contain a mapping for the provided PC.

#### Adding New Entries

Entries can only be added after the decode phase has finished executing. Only then will the instruction be identified as an R-Type (and therefore not requiring an entry in the table), or an I or J-Type. Entries are added into a circular buffer. Once the buffer is full, older entries are purged in a FIFO manner. All new additions are initialized with each saturated counter weakly predicting “not taken”, but exactly one increment away from weakly predicting “taken”.

#### Updating Current Entries

If the entry is found, then the history window is updated to reflect the latest branch taken or not taken information. It also updates the saturated counters as appropriate.

# Register Transfer Language

The fetch phase is excluded so as to avoid unnecessary redundancy in the table. Instead, the fetch phase for all instructions is detailed after the following table. It is also assumed that any relevant data is automatically copied down the pipeline.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **OPC** | **DECODE** | **EXECUTE** | **MEMORY** | **WRITEBACK** |
| NOP | - | - | - | - |
| ADD | D/X\_RS ← F/D\_IRrs,  D/X\_RT ← REG[F/D\_IRrt],  D/X\_RD ← REG[F/D\_IRrd] | X/M­\_ALU ← D/X\_RT + D/X\_RS | - | REG[M/W\_RD] ← M/W\_ALU |
| ADDI | D/X\_RS ← F/D\_IRrs,  D/X\_RT ← REG[F/D\_IRrt],  D/X\_IMM ← F/D\_IRimm | X/M­\_ALU ← D/X\_RT + D/X\_IMM | - | REG[M/W\_RS] ← M/W\_ALU |
| SUB | D/X\_RS ← F/D\_IRrs,  D/X\_RT ← REG[F/D\_IRrt],  D/X\_RD ← REG[F/D\_IRrd] | X/M­\_ALU ← D/X\_RT - D/X\_RS | - | REG[M/W\_RD] ← M/W\_ALU |
| AND | D/X\_RS ← F/D\_IRrs,  D/X\_RT ← REG[F/D\_IRrt],  D/X\_RD ← REG[F/D\_IRrd] | X/M­\_ALU ← D/X\_RT & D/X\_RS | - | REG[M/W\_RD] ← M/W\_ALU |
| OR | D/X\_RS ← F/D\_IRrs,  D/X\_RT ← REG[F/D\_IRrt],  D/X\_RD ← REG[F/D\_IRrd] | X/M­\_ALU ← D/X\_RT | D/X\_RS | - | REG[M/W\_RD] ← M/W\_ALU |
| XOR | D/X\_RS ← F/D\_IRrs,  D/X\_RT ← REG[F/D\_IRrt],  D/X\_RD ← REG[F/D\_IRrd] | X/M­\_ALU ← D/X\_RT ⊕ D/X\_RS | - | REG[M/W\_RD] ← M/W\_ALU |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **OPC** | **DECODE** | **EXECUTE** | **MEMORY** | **WRITEBACK** |
| SLL | D/X\_RS ← F/D\_IRrs,  D/X\_RT ← REG[F/D\_IRrt],  D/X\_RD ← REG[F/D\_IRrd] | X/M­\_ALU ← D/X\_RT << D/X\_RS | - | REG[M/W\_RD] ← M/W\_ALU |
| SRL | D/X\_RS ← F/D\_IRrs,  D/X\_RT ← REG[F/D\_IRrt],  D/X\_RD ← REG[F/D\_IRrd] | X/M­\_ALU ← D/X\_RT >> D/X\_RS | - | REG[M/W\_RD] ← M/W\_ALU |
| SRA | D/X\_RS ← F/D\_IRrs,  D/X\_RT ← REG[F/D\_IRrt],  D/X\_RD ← REG[F/D\_IRrd] | X/M­\_ALU ← D/X\_RT >>a D/X\_RS | - | REG[M/W\_RD] ← M/W\_ALU |
| SB | D/X\_RS ← F/D\_IRrs,  D/X\_RT ← REG[F/D\_IRrt],  D/X\_IMM ← F/D\_IRimm | X/M­\_ALU ← D/X\_RT + D/X\_IMM | MEM[X/M\_ALU] ← X/M\_RS | - |
| LB | D/X\_RS ← F/D\_IRrs,  D/X\_RT ← REG[F/D\_IRrt],  D/X\_IMM ← F/D\_IRimm | X/M­\_ALU ← D/X\_RT + D/X\_IMM | M/W\_MDR ← MEM[X/M\_ALU] | M/W\_RS ← M/W\_MDR |
| BEZ | BPTW­\_PC ← F/D\_PC,  BPTNPC ← EA,  BPTtaken ←RS == 0 | - | - | - |
| BNE | BPTW­\_PC ← F/D\_PC,  BPTNPC ← EA,  BPTtaken ←RS != 0 | - | - | - |
| JUMP | BPTW­\_PC ← F/D\_PC,  BPTNPC ← EA,  BPTtaken ← 1 | - | - | - |
| HALT | - | - | - | - |
|  |  |  |  |  |

## Fetch Phase

The fetch phase is where the branch prediction takes over, and consists of two exclusive possibilities:

1. An R-Type instruction will not cause the branch predictor to engage beyond the identification step. Since all R-Type instructions do not modify the PC or cause any branches in executions, the branch predictor will never override the PC.

IRF/D ← MEM[PC]

PC ← PC + 4

1. In the event of successful identification in the BPT, then the following will happen.

BPTR\_PC ← PC

PC ← BPTPPC