Program Analysis and Transformation

Final Assignment

Benjamin Brandt Ohrt, zpn492

May 29, 2019

Abstract

The Pendulum Instruction Set Architecture (PISA) was first introduced in 1995 by Carlin James Vieri and is a reversible assembly language. The assembly language has later been improved and several high-level languages has been build upon it. One of those languages is the extension to the reversible object-oriented programming language (ROOPL) ROOPL++ presented in 2018. One of the issues with ROOPL++ was the amount of produced target code. In this paper we compile source code for the data structure of a binary tree, analyse the possibility of redundant target code and points out where an optimization could be done.

Contents

1 Introduction

1.1 Assembly language

A CPU understands its own machine language. Instructions in machine language are numbers stored as bytes in memory. Each instruction has its own unique numeric code called its operation code or opcode for short. An example is the instruction that says add EAX and EBX registers together and store the result back into EAX is encoded by the following hex code:

03 C3

This not readable for the blunt eye. Which is why we have a program called an assembler. An assembler is a program that reads the text of a assembly language program and converts the assembly into machine code. [5, p. 11].

The assembly language origins back to Birkbeck College 1946-1962, where the creation was credited to Kathleen Booth.[1] Kathleen Booth was a PhD whom both visited with John Von Neumann with the famous The Von Neumann Architecture[3], helped with the design of multiple machines and was one the founders of the Birkbeck department of computer science.

The assembly language is a symbolic programming language, closest to machine code.[2] An assembly language program is stored as plain text. The text consists of a set of instructions which is processed chronological. Each assembly instruction is equal to one machine instruction for example could above addition be shown as:

add EAX EBX

With the assembly language words as add can be used as mnemonic for the instruction add and so fourth. [5, p. 11].

1.1.1 Example program

```
j main                          # for int i = 0; i < 100; i++
                                #    count += i * 2
loop:
mul $t0, $a1, 2
add $a0, $a0, $t0               # count += i * 2
addi $a1, $a1, 1                # i++
slt $t0, $a1, $a2
beq $t0, 1, loop                # i < 100
jr $ra                          # return

main:
addi $a0, $zero, 0              # count = 0
addi $a1, $zero, 0              # i = 0
addi $a2, $zero, 100            # max
jal loop
```

Small example program written in MIPS assembly

1.1.2 Instruction operands - [OVERVEJ]

Machine code instructions vary, but in general each instruction itself will have fixed number of operands (0 to 3). Operands can have the following types:[5, p. 12]

register: These operands refer directly to the content of the CPU-register.

memory: These refer to data in memory. The address of the data may be a constant hardcoded into the instruction or may be computed using values of registers. Address are always offsets from the beginning of a segment.

immediate: These fixed values that are listed in the instruction itself. They are stored in the instruction itself (in code segment), not in the data segment.

implied: These operands are not explicitly shown. For example the increment instruction adds one to the register or memory. The one is implied.

1.2 Reversible computing

A reversible computing system has, at any time, at most a single previous computation state as well as a single next computation state, and thus a reversible computing system.[6] It comes with the promise of reduced energy consumption, when erasure of information is left out of the program, all that described by Charles Bennet.[7]

1.2.1 Pendulum microprocessor

Pendulum is a reversible microprocessor, invented by Carlin James Vieri at MIT, with an offset in the existing MIPS R2000 architecture.[REF til ARK][4, p. 29] The processor has three registers for used in the control flow:

1 The program counter (PC) for storing adress of the current instruction

2 The branch register (BR) for storing jump offsets

3 The direction bit (DIR) for keeping track of the execution direction.

PC is incremented/decremented by the value stored in DIR, thus changing DIR between 1 & -1, will decide in which direction the instructions will be executed.

1.2.2 Reverse operations

1.2.3

[4, Vieri - Chapter 3, p. 21]

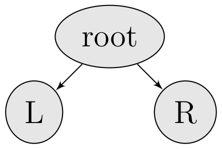
1.3 Motivation

1.4 Outline

2 Data structures

2.1 Binary tree

A binary tree in computer science is a data structure, which is either empty or consist of a root node, where each node has atmost two children. You could see the binary tree as a tuple of the single node root and the two binary trees L & R.



Simple binary tree, with the a root node and two binary trees L & R as children to the root.

Their are different types of binary tree all with their own properties. fx A full binary tree is one where every node either have zero or two child-nodes. A perfect binary tree is one, where all nodes have two children except for the A balanced binary tree is one where L & R of every node only differ in height with no more than one. [REF-Introduction to algorithms]

Properties of binary tree can maximum height, number of nodes

3 Programming languages

3.1 Pendulum Assembly Language

Pendulum Instruction Set Architecture(PISA) is a reversible assembly language, which can assembles by the Pendulum microprocessor.[4, p. 48]

Pendulum instructions is almost identically to the conventional processor, with exception of two constraints:[4, p. 32]

come-from: Must be the predecessor of any jump or branch instruction.

exchange: Memory access is always an exchange. The exchange instruction swaps register value with the value in memory at an address specified by another register.

A simple example of *come-from* is shown below:

```
top:        BRA start   # Branch to start
...                     # Unknown instructions
start:      BRA top     #
```

Pendulum Assembly program

Operation type					
R	op	rsd	rs	sh/rot	func
bit	6	5	5	5	11
J	j/cf	Target			
bit	6	26			
B	j/b op	ra	rb	offset	
bit	6	5	5	16	
I	op	rsd	immediate		
bit	6	5	21		

Jump instructions other than j and all conditional branches uses B-type instruction encoding and specify two registers and an offset.

Immediate instructions, I-type, specify one register and a 21 bit signed or unsigned, depending on the opcode, immediate value.

Instruction	
ADD	
SUB	
ADDI	
AND	
ANDI	
BEQ	
BGEZ	
BGEZAL	
BGTZ	
BLEZ	Branch on less than or equal to zero
BLTZ	Branch on less than zero
BLTZAL	Branch on less than than zero and link
BNE	Branch on not equal
CF	Come-from

3.2 Reversible Object-Oriented Programming

4 Analysis of redundancy

5 Conclusion

6 Further work

References

[1] http://www.computinghistory.org.uk/det/32489/Kathleen-Booth/

[2] https://www.ibm.com/support/knowledgecenter/SSLTBW_2.1.0/com.ibm.zos.v2r1.asma400/asmr102112.htm

[3] https://en.wikipedia.org/wiki/Von_Neumann_architecture

[4] Vieri, C. J. et al. *Pendulum: A Reversible Computer Architecture*. Master's Thesis. University of California at Berkeley 1993.

[5] Carter, P. A. *PC Assembly Language*. 2006

[6] Yokoyama, T. and Glück, R. *A reversible programming language and its invertible self-interpreter*. ACM, 2007.

[7] C. H. Bennet. *Notes on the history of reversible computation*. IBM J. Res. Dev., 32(1), 1988