Department of Computer Engineering,

City College of New York

CSC 342 – Computer Organization [CSC 342_Spring_2023]

DATE: March 1, 2023

Instructor: ***Professor Izidor Gertner***

Student: Paniz Peiravani

Assignment:

Reverse Engineering

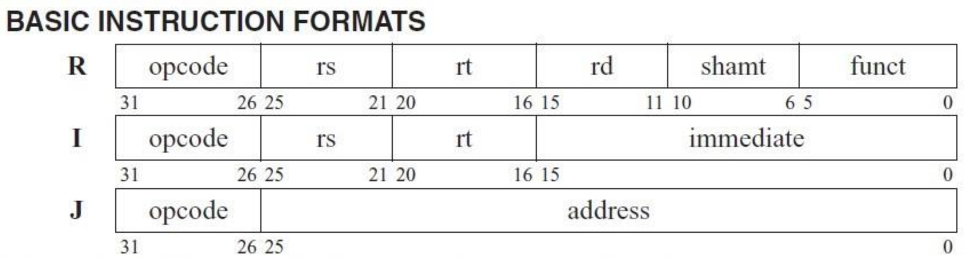
Contents

Objective

The purpose of this assignment is to learn more about reserve engineering. We will summarize reverse engineering from what we learned from the lecture and our textbook chapter 2.5.

Reverse Engineering

Based on MIPS Green page, MIPS arithmetic instructions have three formats which you can see them in *Figure 1*.



BASIC INSTRUCTION FORMATS

R	opcode	rs	rt	rd	shamt	funct
	31 26	25 21	20 16	15 11	10 6	5 0
I	opcode	rs	rt	immediate		
	31 26	25 21	20 16	15 0		
J	opcode	address				
	31 26	25 0				

Figure 1 - Basic Instruction Formats

Here is the meaning of each name of the fields in MIPS instructions:

- **op:** Basic operation of the instruction, traditionally called the opcode.
    - Opcode means the field that denotes the operation and format of an instruction.
- **rs:** The first register source operand.
- **rt:** The second register source operand.
- **rd:** The register destination operand. It gets the result of the operation.
- **shamt:** Shift amount.
- **funct:** Function. This field, often called the function code, selects the specific variant of the operation in the op field.

All instructions in the Architecture are 32 bits in length. There are three different instruction formats: R-Type instructions, I-Type instructions, and J-Type instructions.

R-type (for register) or R-format Instruction Format

31 – 26	25 – 21	20 – 16	15 – 11	10 – 6	5 – 0
op	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

Instruction rd, rs, rt

rs => Source register specifier

rt => Source/Destination register specifier

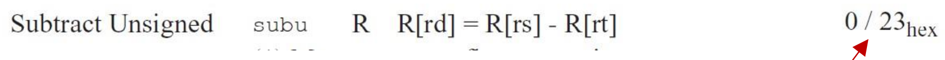
rd => Destination register specifier

An R-Type instruction contains 6 fields: a 6-bit function code (funct), a 5-bit shift amount (shamt), three 5 bit register addresses (rd, rt, rs), and a 6 bit operation code (opcode) which is always zero. In another words, first 6 bits, opcode, always is zero. The next 5 bits is for register specifier, rs, which holds the value of the first operand. The next 5 bits is for register specifier, rt, which holds the value of the second operand. The next 5 bits store the destination register specifier that stores the result of the instruction. The next 5 bits, shift amount, are used for shift instruction instead of rt to make the hardware simpler. Last 6 bits, function, specifies to the hardware exactly R-format instruction to execute.

Example

Imagine we have subu 0x0253a023.

Based on the MIPS green page, we can say that subu is R type.



Subtract Unsigned	subu	R	R[rd] = R[rs] - R[rt]	$0 / 23_{hex}$

Figure 2 - MIPS Green Page

Step One:

Convert hex to 32-bit binary:

- 0x0253a023 = 0000 0010 0101 0011 1010 0000 0010 0011

Step Two:

Break the binary code into groups:

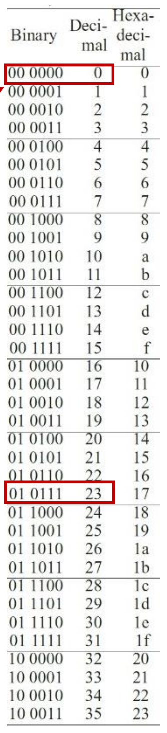
- Opcode | rs | rt | rd | shamt | funct
- 000000 | 10010 | 10011 | 10100 | 00000 | 100011

Step Three:

Since its R instruction the opcode is always zero and the instruction executed is in funct:

By using MIPS green page, we can find the instruction:

- Opcode = 000000 = 0_{hex}
- funct = 100011 = 23_{hex}
- 23_{hex} = subu



Binary	Decimal	Hexadecimal
00 0000	0	0
00 0001	1	1
00 0010	2	2
00 0011	3	3
00 0100	4	4
00 0101	5	5
00 0110	6	6
00 0111	7	7
00 1000	8	8
00 1001	9	9
00 1010	10	a
00 1011	11	b
00 1100	12	c
00 1101	13	d
00 1110	14	e
00 1111	15	f
01 0000	16	10
01 0001	17	11
01 0010	18	12
01 0011	19	13
01 0100	20	14
01 0101	21	15
01 0110	22	16
01 0111	23	17
01 1000	24	18
01 1001	25	19
01 1010	26	1a
01 1011	27	1b
01 1100	28	1c
01 1101	29	1d
01 1110	30	1e
01 1111	31	1f
10 0000	32	20
10 0001	33	21
10 0010	34	22
10 0011	35	23

Figure 3 – MIPS Green Page

Step Four:

Find the rs, rt, and rd specifier:

- rs = 10010 = $1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$ = 18
- rt = 10011 = $1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$ = 19
- rd = 10100 = $1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$ = 20

Step Five:

Since shamt is 00000 we can ignore the shifting.

Step Six:

Combine the assembly instruction.

- subu rd, rs, rt
- subu 20, 19, 18

Step Seven:

Use the MIPS to find the register names used.

18 = $s2

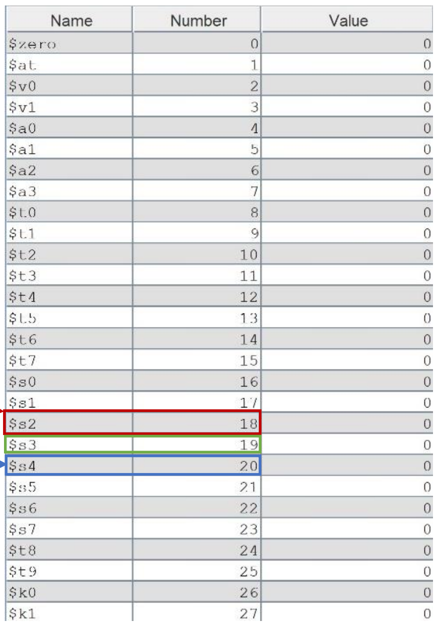
19 = $s3

20 = $s4

rd = 20 = $s4

rs = 19 = $s3

rt = 18 = $s2



Name	Number	Value
$zero	0	0
$at	1	0
$v0	2	0
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	0
$t1	9	0
$t2	10	0
$t3	11	0
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	0
$s1	17	0
$s2	18	0
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0

Figure 4 - MIPS Registers

Final Answer:

opcode rd, rs, rt

- **subu $s4, $s3, $s2**

I-type (for immediate) or I-format Instruction Format

Next instructor that we want to talk about is I type instructor. This format is used by the immediate and data transfer instructions.

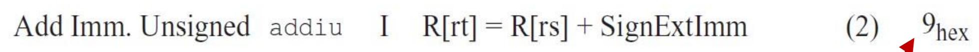
31 – 26	25 – 21	20 – 16	15 – 0 (immediate)
op	rs	rt	constant or address
6 bits	5 bits	5 bits	16 bits

First 6 bits are opcode which represent the type of the instruction that will be executed. The next 5bits, rs, are the source register specifier. The next 5 bits are the rt, which is target register specifier. Last 16 bits are used for immediate value which depending on the value it can be a sign extended or zero extended.

Example

Imagine we have addiu 0x2412000e.

Based on the MIPS green page, we can say that addiu is I type



Add Imm. Unsigned addiu I R[rt] = R[rs] + SignExtImm (2) 9_{hex}

Figure 5 - MIPS Green Page

Step One:

Convert hex to 32-bit binary:

- 0x2412000e = 0010 0100 0001 0010 0000 0000 0000 1110

Step Two:

Break the binary code into groups:

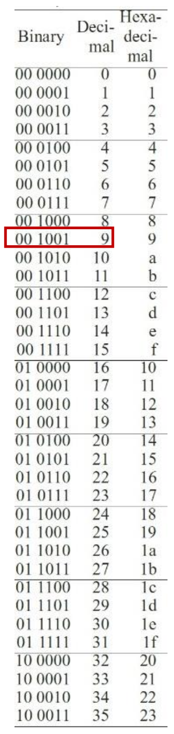
- Opcode | rs | rt | immediate
- 001001 | 00000 | 10010 | 0000000000001110

Step Three:

Find which instruction using MIPS green page.

By using MIPS green page, we can say:

- Opcode = 001001 = 9_{hex}
- 9_{hex} = addiu



Binary	Deci-mal	Hexa-deci-mal
00 0000	0	0
00 0001	1	1
00 0010	2	2
00 0011	3	3
00 0100	4	4
00 0101	5	5
00 0110	6	6
00 0111	7	7
00 1000	8	8
00 1001	9	9
00 1010	10	a
00 1011	11	b
00 1100	12	c
00 1101	13	d
00 1110	14	e
00 1111	15	f
01 0000	16	10
01 0001	17	11
01 0010	18	12
01 0011	19	13
01 0100	20	14
01 0101	21	15
01 0110	22	16
01 0111	23	17
01 1000	24	18
01 1001	25	19
01 1010	26	1a
01 1011	27	1b
01 1100	28	1c
01 1101	29	1d
01 1110	30	1e
01 1111	31	1f
10 0000	32	20
10 0001	33	21
10 0010	34	22
10 0011	35	23

Figure 6 – MIPS Green Page

Step Four:

Find the rs and rt specifiers:

- rs = 00000 = 0
- rt = 10010 = $1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 18$

Step Five:

Find immediate.

- 0000 0000 0000 1110 = $5_{hex} = 14_{dec}$

Step Six:

Combine the assembly instruction.

- addiu rs, rt, immediate
- addiu 0, 18, 5

Step Seven:

Use the MIPS to find the register names used.

0 = $zero

18 = $s2

rs = 0 = $zero

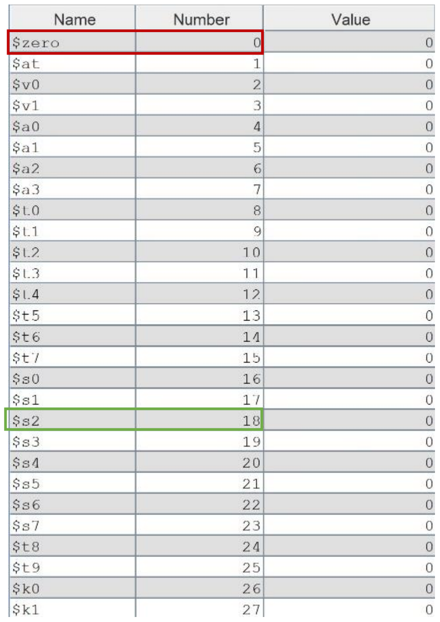
rt = 18 = $s2

14 dec = 0x0000000e

Final Answer:

opcode rt, rs, immediate constant

- **addiu $s2, $zero, 14**
- **addiu $s2, $zero, 0x0000000e**



Name	Number	Value
$zero	0	0
$at	1	0
$v0	2	0
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	0
$t1	9	0
$t2	10	0
$t3	11	0
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	0
$s1	17	0
$s2	18	0
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0

Figure 7 - MIPS Registers

Conclusion

This assignment's goal was to teach us how to think as an inverse engineer. It helped us to learn how to find the MIPS instruction from hexadecimal by using the seven steps the using MIPS Green page.