Tutorial 2
CSE 112 Computer Organization

Topic:
We will analyze a subset of R-type instructions available in the RISC-V ISA

Semantic	Explanation
add rd, rs1, rs2	***Add*** Add the content of the rs1 and rs2 and store the result in rd. rd = rs1 + rs2
sub rd, rs1 rs2	***Subtract*** Two's complement subtraction. rd = rs1 - rs2.
sll rd, rs1, rs2	***Shift left logical*** rd = rs1 << unsigned(rs2[4:0]) Left shift rs1 by the value in lower 5 bits of rs2.
Note: Appending zeros at required places is called logical shift.	
slt rd, rs1, rs2	***Set if less than*** rd=1. only if signed(rs1) < signed(rs2)
Note: Here we treat the register content of rs1, rs2 as signed integers.	
sltu rd, rs1, rs2	***Set if less than unsigned*** rd=1. Only if unsigned(rs1) < unsigned(rs2)
Note: Here we treat the register content of rs1, rs2 as unsigned integers.	
xor rd, rs1, rs2	***Bitwise Xor*** rd=rs1 ⊕ rs2
srl rd, rs1, rs2	***Shift right logical*** rd=rs1 << unsigned(rs2[4:0]) Left shift rs1 by the value in lower 5 bits of rs2.
Note: Here no sign is preserved.	
or rd, rs1, rs2	***Bitwise Or*** rd=rs1 | rs2
and rd, rs1, rs2	***Bitwise And*** rd=rs1 & rs2

R-type instruction:

Format:

funct7	rs2	rs1	funct3	rd	opcode
[31:25]	[24:20]	[19:15]	[14:12]	[11:7]	[6:0]

funct7	rs2	rs1	funct3	rd	opcode	Instruction
[31:25]	[24:20]	[19:15]	[14:12]	[11:7]	[6:0]	
0000000			000		0110011	add
0100000			000		0110011	sub
0000000			001		0110011	sll
0000000			010		0110011	slt
0000000			011		0110011	sltu
0000000			100		0110011	xor
0000000			101		0110011	srl
0000000			110		0110011	or
0000000			111		0110011	and

Note: All registers in the above-mentioned ISA are 32-bit. Furthermore, the register **"zero"** is hardwired to **zero**. Any read from this register provides zero; the write to this will remain unaffected.

Q1. What is an ISA?

Q2. The table is a subset of R-type Instructions available in the RISC-V ISA. We need to answer the questions below.

1. How many registers are there in the ISA per the mentioned R-type subset?
2. What is the minimum and maximum signed and unsigned value that can be stored in any register?
3. How many unique operations are possible in the mentioned R-type subset of RISC-V?
4. What is the size of each instruction in bits?
5. What is the role of funct3 and funct7 and opcode in each mentioned instruction type?
6. Why do "**add**" and **"sub"** have different **func7** and the rest have the same value of **funt3**?
7. What are the drawbacks and benefits of having more number of addressable registers as per ISA?
8. Which logical or bitwise operation would you like to add as a custom instruction in ISA?
9. Find the value of r3 after the execution of **every** below-mentioned instruction
    a. r1=0x0000_0000; r2=0x8000_0000;

i. add r3, r2, **zero**
ii. add **zero**, r1, **zero**
iii. add r3, r2, r1

b. r1=0x8000_0000; r2=0x8000_0000;
i. add r3, r2, r1

c. r1=0x8000_0000; r2=0x8000_0001;
i. sub r3 r2, r1

d. r1=0x0000_0030; r2=0x0000_0040;
i. sub r3 r2, r1

e. r1=0x0002_0701; r2=0x0000_0010;
i. sll r3, r1, r2

f. r1=0x0002_0701; r2=0x0000_000A;
i. sll r3, r1, r2

g. r1=0x8000_0000; r2=0x8000_0001;
i. slt r3, r2, r1
ii. slt r3, r1, r2

h. r1=0x8000_0000; r2=0x8000_0001;
i. sltu r3, r2, r1
ii. sltu r3, r1, r2

i. r1=0x0C0C_0000; r2=0xC0C0_0101;
i. xor r3, r2, r1

j. r1=0x0002_0701; r2=0x0000_000A;
i. srl r3, r1, r2

k. r1=0x0C0C_0000; r2=0xC0C0_0101;
i. or r3, r2, r1

l. r1=0x0C0C_0000; r2=0xC0C0_0101;
i. and r3, r2, r1

Note: The notation “0x0_” represents the alphabets trailing the “x” characters are hexadecimal and “_” is added just to improve the readability.

Remark: Here is a link to look for the RISC-V ISA with a good explanation RISC-V.

Solution:

Q1. ISA- The semantics of all the instructions supported by a processor is known as its instruction set architecture (ISA). This includes the semantics of the instructions themselves, along with their operands and interfaces with peripheral devices.

Q2

1. As we have only 5-bits to address each register uniquely. So, only 32 registers (general purpose) registers are available as per the above-mentioned subset of ISA.
2. As each register is of 32 bit.
    a. The maximum positive value is 2^(32-1) - 1. Or 2^{32-1} - 1.
    b. The minimum value or maximum negative value is - 2^(32-1).
3. We can differentiate the operations on the basis of funct3, funct7. This is a little vague statement as the value of funct3 and funct7 are mapped to some type of instructions in the RISC-V ISA.
    a. On the basis of funct3 (3-bit) we can have 8 possible unique registers.

4. As per the complete instruction code format given. The size of each instruction is 32-bit.
5. Role of funct3, funct7, opcode.
    a. To programmers they are used to distinguish among the instructions.
    b. To Microprocessor designers they are utilized as to create efficient hardware (little vague).
        i. For example The **“add”** and **“sub”** instructions have the same **funct3** and **opcode** value but different values of **funct7**. They are using a single adder and this bit is used as a flag to perform or not perform 2’s complement before addition.
6. Already answered in question 5.
7. It is always beneficial to have more number of general purpose registers. As, they can be used to store data inside the microprocessor and perform less or minimal communication with cache. But, each register needs a unique address to address itself.
    a. For example 32 registers can be addressed with 5-bits only. And 64 registers need 6-bit of address for each. This will increase the instruction size in bits or we need to reduce the number of unique operations (if keeping the instruction size same).
8. Please answer this.
9. The values of r3 after the execution of corresponding instructions.
    a. r1=0x0000_0000; r2=0x8000_0000;
        i. r3=0x8000_0000;
        ii. r3=”unpredictable”. But the instruction executes properly.
        iii. r3=0x8000_0000;
    b. r1=0x8000_0000; r2=0x8000_0000;
        i. r3=0x0000_0000; Arithmetic overflow is ignored.
    c. r1=0x8000_0000; r2=0x8000_0001;
        i. r3=0x0000_0001; Arithmetic overflow is ignored.
    d. r1=0x0000_0030; r2=0x0000_0040;
        i. r3=0x0000_0010
    e. r1=0x0002_0701; r2=0x0000_0010;
        i. r3=0x0701_0000
    f. r1=0x0002_0701; r2=0x0000_000A;
        i. r3=0x081C_0400
    g. r1=0x8000_000; r2=0x8000_0001;
        i. r3=0x0000_00001
        ii. r3=0x0000_0000
    h. r1=0x8000_000; r2=0x8000_0001;
        i. r3=0x0000_0000
        ii. r3=0x0000_0001
    i. r1=0x0C0C_0000; r2=0xC0C0_0101;
        i. r3=0xCCCC_0101
    j. r1=0x0002_0701; r2=0x0000_000A;
        i. r3=0x0000_0101
    k. r1=0x0C0C_0000; r2=0xC0C0_0101;
        i. r3=0xCCCC_0101
    l. r1=0x0C0C_0000; r2=0xC0C0_0101;
        i. r3=0x0000_0000;