1.5.9 Exercise 9

1. Provide the instruction type, assembly language instruction, and binary representation of instruction described by the following RISC-V fields:

   - opcode=0x33, funct3=0x0, funct7=0x20, rs2=5, rs1=7, rd=6
   - opcode=0x3, funct3=0x3, rs1=27, rd=3, imm=0x4

2. Considering the standard 32-bit RISC-V instruction formats, convert the following instructions to machine code.

   - add x1, x2, x3
   - addi x1, x2, 100
   - lb x1, 4(x2)
   - beq x6, x8, 1024
   - lw t5, 17(t6)

3. What are the instructions encoded by:

   - 0x003122B3
   - 0x00820783
   - 0x00219223
   - 0xEE151917

4. Consider the following assembly code:

   ```
   .text
   1. mv s1 a0
   2. addi s2 s2 4
   3. Start: beq s1 x0 End
   4. lw a0 0(s1)
   5. jal ra printf
   6. add s1 s2 s1
   7. lw s1 0(s1)
   8. jal x0 Start
   9. End: jalr x0, ra, 0
   ```

   a. What is the machine code generated for beq s1 x0 End (line 3).
   b. Given the hex representation, which line number in the above program does it correspond to?
      0x0004A483
      0xFEDFF06F

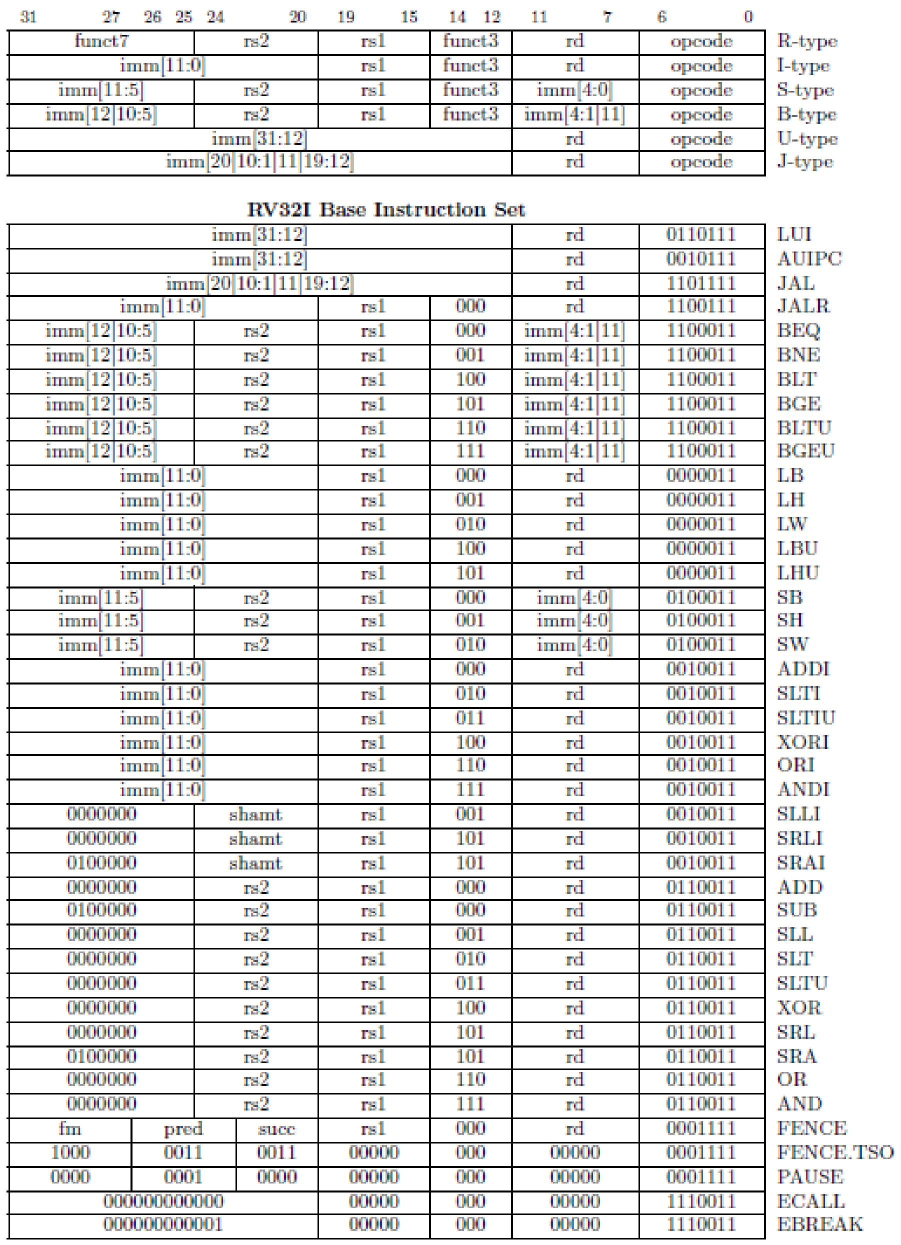
5. You want to build a mini RISC-V instruction architecture that only supports 16 registers, which allows the length of the register fields to be shortened. Assuming that you use the extra bits to extend the immediate field, what is the range of half-word instructions that can be reached using a branch instruction in this new format? [<lower bound>, <upper bound>]

1.5.10 Exercise 10

When the execution reaches point Break!, what location will the five expressions: &year, Name, Game, Ver, &ver give? (i.e. “stack”, “heap”, “static” or “text”)

```
#include <stdio.h>
#include <stdlib.h>
static const int year = 2019;

int main (void) {
char name[] = "Jose";
char *game = "The Elder Scrolls";
int *ver = malloc(sizeof(int));
*ver = 5;
/* Break! */
printf("Until %d, %s's favourite game is %s %d.", year, name, game, *ver);
return 0;
}
```



31 27 26 25	24 20	19 15	14 12	11 7	6 0	
funct7	rs2	rs1	funct3	rd	opcode	R-type
imm[11:0]		rs1	funct3	rd	opcode	I-type
imm[11:5]	rs2	rs1	funct3	imm[4:0]	opcode	S-type
imm[12|10:5]	rs2	rs1	funct3	imm[4:1|11]	opcode	B-type
imm[31:12]				rd	opcode	U-type
imm[20|10:1|11|19:12]				rd	opcode	J-type

RV32I Base Instruction Set

imm[31:12]				rd	0110111	LUI
imm[31:12]				rd	0010111	AUIPC
imm[20|10:1|11|19:12]				rd	1101111	JAL
imm[11:0]		rs1	000	rd	1100111	JALR
imm[12|10:5]	rs2	rs1	000	imm[4:1|11]	1100011	BEQ
imm[12|10:5]	rs2	rs1	001	imm[4:1|11]	1100011	BNE
imm[12|10:5]	rs2	rs1	100	imm[4:1|11]	1100011	BLT
imm[12|10:5]	rs2	rs1	101	imm[4:1|11]	1100011	BGE
imm[12|10:5]	rs2	rs1	110	imm[4:1|11]	1100011	BLTU
imm[12|10:5]	rs2	rs1	111	imm[4:1|11]	1100011	BGEU
imm[11:0]		rs1	000	rd	0000011	LB
imm[11:0]		rs1	001	rd	0000011	LH
imm[11:0]		rs1	010	rd	0000011	LW
imm[11:0]		rs1	100	rd	0000011	LBU
imm[11:0]		rs1	101	rd	0000011	LHU
imm[11:5]	rs2	rs1	000	imm[4:0]	0100011	SB
imm[11:5]	rs2	rs1	001	imm[4:0]	0100011	SH
imm[11:5]	rs2	rs1	010	imm[4:0]	0100011	SW
imm[11:0]		rs1	000	rd	0010011	ADDI
imm[11:0]		rs1	010	rd	0010011	SLTI
imm[11:0]		rs1	011	rd	0010011	SLTIU
imm[11:0]		rs1	100	rd	0010011	XORI
imm[11:0]		rs1	110	rd	0010011	ORI
imm[11:0]		rs1	111	rd	0010011	ANDI
0000000	shamt	rs1	001	rd	0010011	SLLI
0000000	shamt	rs1	101	rd	0010011	SRLI
0100000	shamt	rs1	101	rd	0010011	SRAI
0000000	rs2	rs1	000	rd	0110011	ADD
0100000	rs2	rs1	000	rd	0110011	SUB
0000000	rs2	rs1	001	rd	0110011	SLL
0000000	rs2	rs1	010	rd	0110011	SLT
0000000	rs2	rs1	011	rd	0110011	SLTU
0000000	rs2	rs1	100	rd	0110011	XOR
0000000	rs2	rs1	101	rd	0110011	SRL
0100000	rs2	rs1	101	rd	0110011	SRA
0000000	rs2	rs1	110	rd	0110011	OR
0000000	rs2	rs1	111	rd	0110011	AND
fm	pred succ	rs1	000	rd	0001111	FENCE
1000	0011 0011	00000	000	00000	0001111	FENCE.TSO
0000	0001 0000	00000	000	00000	0001111	PAUSE
000000000000		00000	000	00000	1110011	ECALL
000000000001		00000	000	00000	1110011	EBREAK