Question 1 (20 points): You've been hired to create instruction formats for MIPS-48, a new version of MIPS that has 48 32-bit registers instead of 32 registers. These registers will be numbered 0 to 47, instead of 0 to 31. Assume that 0-31 are the original MIPS registers and 32-47 are the new registers. Other than the number of registers, MIPS-48 is just like MIPS.

Part A (2 points): How many bits are required to represent a register in a MIPS-48 instruction?

In 5 bits, we can only represent 32 things.

Therefore, we need 6 bits to represent the 48 registers of MIPS-48.

Part B (8 points): MIPS-48 has 32-bit instructions. Assume that the opcode field remains the same, and that for I-type instructions, extra bits needed for the register fields come from the immediate field. What is the highest possible address of the next instruction if the current instruction is **beq** and is at address 0x4000 0000?

32 bits − (6 bits for opcode + 2× 6 bits for registers) = 14 bits for immediate.

The largest positive signed int in 14 bits is 01 1111 1111 1111 ⇒ 0x1FFF

Shift this left by 2 to get number of bytes ⇒ 0x7FFC

Add this offset to the original address +4 : 0x4000 0000 + 7FFC + 4 = 0x4000 8000

0x4000 8000

Part C (6 points): Give the hexadecimal representation of the MIPS-48 instruction **beq $s0, $s1, foo**, if that instruction is at 0x4000 0000 and the label **foo** refers to an instruction at address 0x4000 00A8.

The offset is: (target addr − dest addr − 4) shifted by 2

opcode = 0x4 (6 bits)
rs = $s0 = 0x10 (6 bits)
rt = $s1 = 0x11 (6 bits)

0xA8 − 4 = 0xA4 ⇒ (shift) 0x29 (14 bits)

000100 | 01 0000 | 01 0001 | 00 0000 0010 1001 = **0x1104 4029**

opcode | rs | rt | immediate

Part D (2 points): How many more registers could be added to MIPS-48 without changing the opcode field or affecting the range of a branch instruction?

6 bits can represent 64 things (values 0-63)
MIPS-48 uses 48, so could add 16 more.

Part E (2 points): What would the format for a J-type instruction be in MIPS-48?

There are no registers in a J-type instruction, so it would look the same as MIPS.

ie.