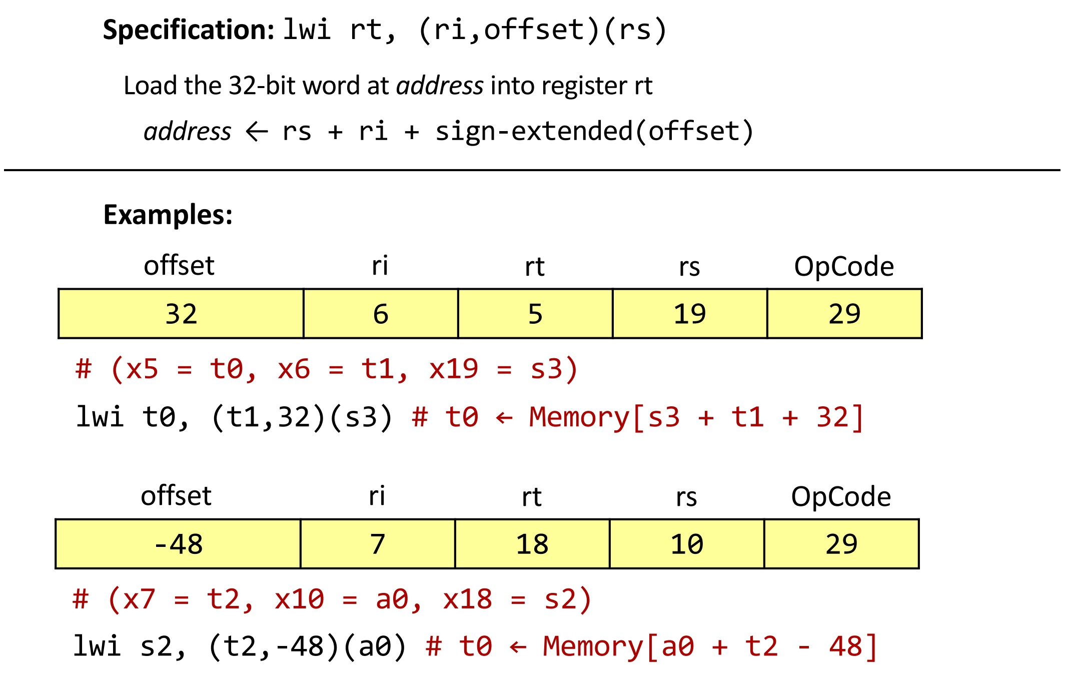
# Load Word Indirect

**Question 1:** (5 points)
(V01, V02, V05) After hearing from many programmers and compiler developers that a register-indirect register addressing mode would reduce the number of instructions executed by RISC-V processors, a maker for RISC-V processors designed a new version of the processor with a new set of load instructions that they identified by adding the letter `i` to the name of the instruction to indicate that it is using an indirect addressing mode. For instance, the specification for the *load word indirect* `lwi` and two examples illustrating the binary format for this new instruction are shown in Figure **??**. Notice that they decided to use the opcode 29 (expressed in base 10 here) for the `lwi` instruction. Recall that opcodes in the RISC-V Instruction Set Architecture are formed by seven bits.



**Specification:** `lwi rt, (ri,offset)(rs)`

Load the 32-bit word at *address* into register rt

*address* ← `rs + ri + sign-extended(offset)`

**Examples:**

| offset | ri | rt | rs | OpCode |
|---|---|---|---|---|
| 32 | 6 | 5 | 19 | 29 |

```
# (x5 = t0, x6 = t1, x19 = s3)
lwi t0, (t1,32)(s3) # t0 ← Memory[s3 + t1 + 32]
```

| offset | ri | rt | rs | OpCode |
|---|---|---|---|---|
| -48 | 7 | 18 | 10 | 29 |

```
# (x7 = t2, x10 = a0, x18 = s2)
lwi s2, (t2,-48)(a0) # t0 ← Memory[a0 + t2 - 48]
```

Figure 1: Specification and two examples for the new *load word indirect* `lwi` instruction.

**Question 2:** (5 points)
How many bits are used for the offset field in the `lwi` instruction?

**Question 3:** (5 points)
Using your answer from Question 2, what is the largest positive value that the offset may have. Provide your answer both expressed as a decimal value and as a 32-bit number expressed in hexadecimal.

**Question 4:** (5 points)
Using your answer from Question 2, what is the most negative value that the offset may have. Provide your answer both expressed as a decimal value and as a 32-bit number expressed in hexadecimal.

**Question 5:** (5 points)
This new version of the RISC-V processor fetched an instruction from memory whose hexadecimal representation is `0xE201 C91D`. Based on the specification and examples shown in Figure **??**, what is the assembly representation of this instruction? In other words, how would this instruction look like in an assembly program? You must use `t` and `s` registers when writing your final answer.

**Question 6:** (5 points)
Assume that before the execution of the instruction `lwi` specified in Question 5 of this question, the state of the processor is as shown in Table **??**. What is the memory address, expressed in hexadecimal, accessed by that instruction?

| Register | Value | Register | Value |
|---|---|---|---|
| t0 | 0xFFFF FFFC | s0 | 0x0000 00FC |
| t1 | 0xFF00 FF00 | s1 | 0xFFFF FFCC |
| t2 | 0xFEC0 0000 | s2 | 0xFFFF F000 |
| t3 | 0x0AC0 0080 | s3 | 0xFFFF FFEE |
| t4 | 0x0000 4000 | s4 | 0x0000 7FC0 |
| t5 | 0x0008 7400 | s5 | 0x0000 0044 |
| t6 | 0x0010 0100 | s6 | 0x0000 0FF0 |

Table 1: Processor State before the execution of the instruction `lwi`

**Question 7:** (5 points)
What is the binary representation, expressed in hexadecimal, for the following assembly instruction:

```
lwi   s0, (t0,-64)(t1)
```