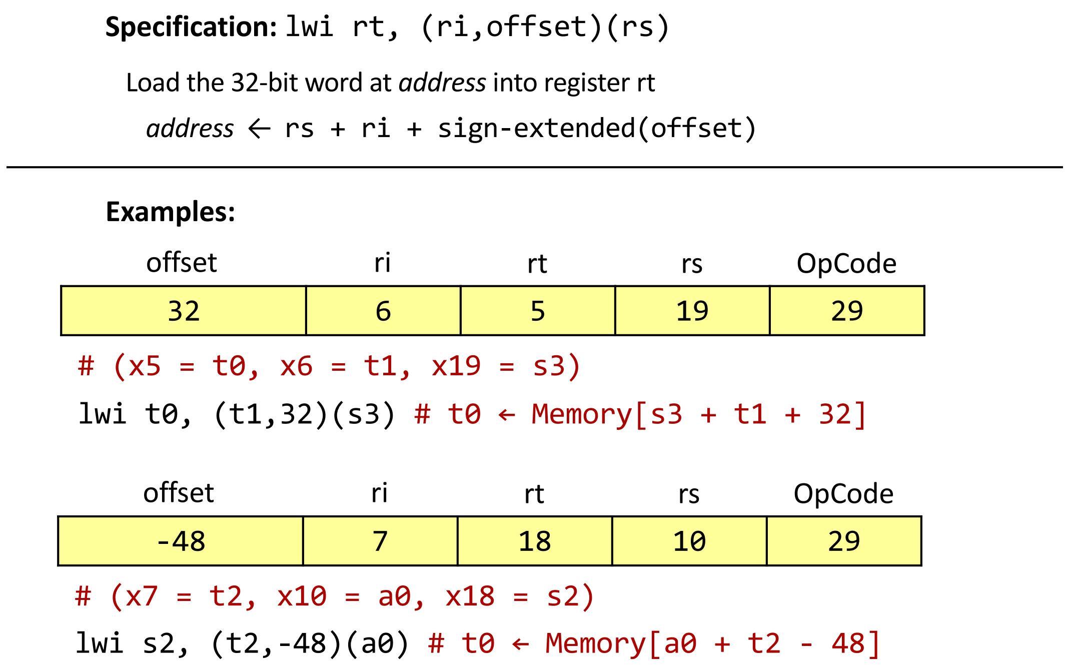
▶Solution◀

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 1. 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? There are 32 registers in RISC-V, thus we need five bits to specify the value of each register, then we need seven

bits for the opcode, thus 7+5+5+5 = 22 bits are used for the registers plus opcode and the remaining 32-22 = 10 bits are used for the offset.

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. In binary the largest positive value is:

```
0000 0000 0000 0000 0000 0001 1111 1111
```

Which is equal $2^9 - 1 = 512 - 1 = 511$ In hexadecimal:

```
0x 0000 01FF
```

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.

In binary the largest negative value is:

```
1111 1111 1111 1111 1111 1110 0000 0000
```

Which is equal $-2^9 = -512$ In hexadecimal:

```
0x FFFF FE00
```

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 1, 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.

The binary representation of the instruction is:

```
1110 0010 0000 0001 1100 1001 0001 1101
```

Separating into the instruction fields we have:

```
offset      ri     rt     rs     Opcode
1110001000  00000  11100  10010  0011101
```

From Figure 1 or the reference sheet: `x0 = zero`, `x28 = t3`, `x18 = s2`. The decimal value of the offset is computed as follows:

$$\begin{aligned}
\text{offset} &= 1110001000 \\
\overline{\text{offset}} &= 0001110111 \\
\overline{\text{offset}} + 1 &= 0001111000 = 2^7 - 2^3 = 128 - 8 = 120 \\
\text{offset} &= -120
\end{aligned}$$

Thus, the assembly code for the instruction is:

```
lwi  t3, (zero,-120)(s2)
```

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 1. What is the memory address, expressed in hexadecimal, accessed by that instruction?

The memory address is given by the value in `s2` plus the sign-extended offset:

```
  0xFFFF F000   = s2
+ 0xFFFF FF88   = sign-extended(offset)
--------------------------------
  0xFFFF EF88
```

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)
```

From Figure 1 `s0 = x8`, `t0 = x5`, `t1 = x6`

The binary representation of -64 in 10 bits is:

$$\begin{aligned} +64 &= 00\ 0100\ 0000 \\ \overline{+64} &= 11\ 1011\ 1111 \\ -64 &= 11\ 1100\ 0000 \end{aligned}$$

```
offset      ri     rt     rs     Opcode
1111000000  00101  01000  00110  0011101
```

Grouping into groups of four bits:

```
1111 0000 0000 1010 1000 0011 0001 1101
```

Thus, the hexadecimal representation is `0xF00A 831D`