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

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

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

Examples:

OpCode	rs	rt	ri	offset
29	19	8	9	32

```
# (R8 = $t0, R9 = $t1, R19 = $s3)
lwi $t0, ($t1,32)($s3) # $t0 ← Memory[$s3 + $t1 + 32]
```

OpCode	rs	rt	ri	offset
29	17	18	16	-48

```
# (R16 = $s0, R17 = $s1, R18 = $s2)
lwi $s2, ($t0,-48)($s1) # $s2 ← Memory[$s1 + $t0 - 48]
```

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

Question 2 (20 points): After hearing from many programmers and compiler developers that a register-indirect register addressing mode would reduce the number of instructions executed by MIPS processors, the makers of the MIPS processor 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 MIPS Instruction Set Architecture are formed by six bits.

a. (**3 points**) How many bits are used for the offset field in the `lwi` instruction?

There are 32 registers in MIPS, thus we need five bits to specify the value of each register, then we need six bits for the opcode, thus 6+5+5+5 = 21 bits are used for the registers plus opcode and the remaining 32-21 = 11 bits are used for the offset.

b. (**3 points**) Using your answer from (a), 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 0011 1111 1111`

Which is equal $2^{10} - 1 = 1024 - 1 = 1023$

In hexadecimal:

`0x 0000 03FF`

c. (**3 points**) Using your answer from (a), 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 1100 0000 0000`

Which is equal $-2^{10} = -1024$

In hexadecimal:

`0x FFFF FC00`

d. (**5 points**) This new version of the MIPS processor fetched an instruction from memory whose hexadecimal representation is `0x7511 0710`. 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:

`0111 0101 0001 0001 0000 0111 0001 0000`

Separating into the instruction fields we have:

```
Opcode  rs    rt    ri    offset
011101  01000 10001 00000 11100010000
```

From Figure 1 or the reference sheet: `R8 = $t0`, `R17 = $s1`, `R0 = $0`.

The decimal value of the offset is computed as follows:

$$\begin{aligned} \text{offset} &= 11100010000 \\ \overline{\text{offset}} &= 00011101111 \\ \overline{\text{offset}} + 1 &= 00011110000 = 2^8 - 2^4 = 256 - 16 = 240 \\ \text{offset} &= -240 \end{aligned}$$

Thus, the assembly code for the instruction is:

```
lwi  $s1, ($0,-240)($t0)
```

e. (**3 points**) Assume that before the execution of the instruction `lwi` specified in item (d) 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 `$t0` plus the sign-extended offset:

```
  0xFFFF F000   = $t0
+ 0xFFFF FF10   = sign-extended(offset)
--------------------------------
  0xFFFF EF10
```

Register	Value	Register	Value
$t0	0xFFFF F000	$s0	0x0000 00FC
$t1	0x FF00 FF00	$s1	0xFFFF FFCC
$t2	0x FEC0 0000	$s2	0xFFFF FFFC
$t3	0x 0AC0 0080	$s3	0xFFFF FFEE
$t4	0x 0000 4000	$s4	0x0000 7FC0
$t5	0x 0008 7400	$s5	0x0000 0044
$t6	0x 0010 0100	$s6	0x0000 0FF0
$t7	0x 0000 DC00	$s7	0x0000 0000

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

f. (**3 points**) What is the binary representation, expressed in hexadecimal, for the following assembly instruction:

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

From Figure 1 `$s0 = R16`, `$t0 = R8`, `$t1 = R9`

The binary representation of -64 in 11 bits is:

$$\begin{aligned} +64 &= 000\ 0100\ 0000 \\ \overline{+64} &= 111\ 1011\ 1111 \\ -64 &= 111\ 1100\ 0000 \end{aligned}$$

```
Opcode  rs    rt    ri    offset
011101  01001 10000 01000 11111000000
```

Grouping into groups of four bits:

```
0111 0101 0011 0000 0100 0111 1100 0000
```

Thus, the hexadecimal representation is `0x753047C0`