**Example**

Instruction: OR $s7, $a3, $t5, each type of instruction (R-type, I-type, J-type)

* Opcode = 0 (hex) = 000000 (binary)
* $rs = $a3= 7 (decimal) = 00111 (binary)
* Srt = $t5 = 21 (decimal) = 10101 (binary)
* $rd = $s7 = 23 (decimal) = 10111 (binary)
* shamt = Not used = 00000 (binary)
* funct = 25(hex) = 010101 (binary)

Binary: 000000 00111 10101 10111 00000 010101

Grouping by 4 bits: 0000 0000 0111 1010 1101 1100 0000 1010 0101

|  |  |  |
| --- | --- | --- |
| Decimal | Hex | Binary |
| 0 | 0 | 0000 |
| 1 | 1 | 0001 |
| 2 | 2 | 0010 |
| 3 | 3 | 0011 |
| 4 | 4 | 0100 |
| 5 | 5 | 0101 |
| 6 | 6 | 0110 |
| 7 | 7 | 0111 |
| 8 | 8 | 1000 |
| 9 | 9 | 1001 |
| 10 | A | 1010 |
| 11 | B | 1011 |
| 12 | C | 1100 |
| 13 | D | 1101 |
| 14 | E | 1110 |
| 15 | F | 1111 |

**Hexadecimal: 0x007D7B85 (0x00F5B813 doesn’t make sense in the homework example)**

**Convert the following MIPS instructions to hexadecimal:**

1. **add $t5, $s1, $s2**

* Opcode: 0 (R-type instruction)
* $rs: $s1 = 17 (decimal) = 10001 (binary)
* $rt: $s2 = 18 (decimal) = 10010 (binary)
* $rd: $t5 = 13 (decimal) = 01101 (binary)
* shamt: 0 (not used) = 00000 (binary)
* funct: 32 (hex) for add = 100000 (binary)

Binary: 000000 10001 10010 01101 00000 100000

Grouping by 4 bits: 0000 0010 0011 0010 0110 1000 0010 0000  
**Hexadecimal: 0x02326820**

1. **lw $t2, 1024($s5)**

The lw instruction is an I-type instruction.

* Opcode: 35 (decimal) = 100011 (binary)
* $rs: $s5 = 21 (decimal) = 10101 (binary)
* $rt: $t2 = 10 (decimal) = 01010 (binary)
* Immediate: 1024 = 0000010000000000 (binary, 16 bits)

Binary: 100011 10101 01010 0000010000000000

Grouping by 4 bits: 1000 1110 1010 1010 0000 0100 0000 0000

**Hexadecimal: 0x8EAA0400**

1. **addi $t1, $s4, 16 or addi $t1, $s4, 0x0010**

The addi instruction is also an I-type instruction.

* Opcode: 8 (decimal) = 001000 (binary)
* $rs: $s4 = 20 (decimal) = 10100 (binary)
* $rt: $t1 = 9 (decimal) = 01001 (binary)
* Immediate: 16 = 0000000000010000 (binary, 16 bits)

Binary: 001000 10100 01001 0000000000010000

Grouping by 4 bits: 0010 0010 1000 1001 0000 0000 0001 0000  
**Hexadecimal: 0x22890010**

1. **sll $t6, $s0, 2 or sll $t6, $s0, 0x2**

The sll instruction is an R-type instruction.

* Opcode: 0 (since this is an R-type instruction) = 000000 (binary)
* $rs: (not used) = 00000 (binary)
* $rt: $s0 = 10000 (binary)
* $rd: $t6 = 14 (decimal) = 01110 (binary)
* shamt: $s0 (shift two) 2 = 00010 (binary)
* funct: 0 (hex) for sll = 000000 (binary)

Binary: 000000 00000 00000 01110 00010 000000

Grouping by 4 bits: 0000 0000 0000 0000 0111 0000 1000 0000

**Hexadecimal: 0x00107080**

1. **nor $t4, $s3, $zero**

The nor instruction is an R-type instruction.

Opcode: 0 (since this is an R-type instruction)

* $rs: $s3 = 19 (decimal) = 10011 (binary)
* $rt: $zero = 0 (decimal) = 00000 (binary)
* $rd: $t4 = 12 (decimal) = 01100 (binary)
* shamt: 0 (not used) = 00000 (binary)
* funct: 39 (hex) for nor = 100111 (binary)

Binary: 000000 10011 00000 01100 00000 100111

Grouping by 4 bits: 0000 0010 0110 0000 0110 0000 0010 0111

**Hexadecimal: 0x02606027**

**Convert the following machine language instructions to MIPS assembly:**

1. **0000 0010 0000 1111 0010 1000 0010 0010**

Opcode: 000000 (binary) = 0, R-type instruction

$rs: 10000 (binary) = 16 (decimal) = $s0

$rt: 01111 (binary) = 15 (decimal) = $t7

$rd: 00101 (binary) = 5 (decimal) = $a1

shamt: 00001 (binary) = 1

funct: 000010 (binary) = 2, which corresponds to srl (shift right logical).

**Instruction: srl $a1, $s0, 1**

1. **0000 0001 0100 1011 0110 0000 0010 0100**

Opcode: 000000 (binary) = 0, R-type instruction

$rs: 01010 (binary) = 10 (decimal) = $t2

$rt: 01011 (binary) = 11 (decimal) = $t3

$rd: 01100 (binary) = 12 (decimal) = $t4

shamt: 00000 (binary) = 0

funct: 100100 (binary) = 36, which corresponds to and.

**Instruction: and $t4, $t2, $t3**