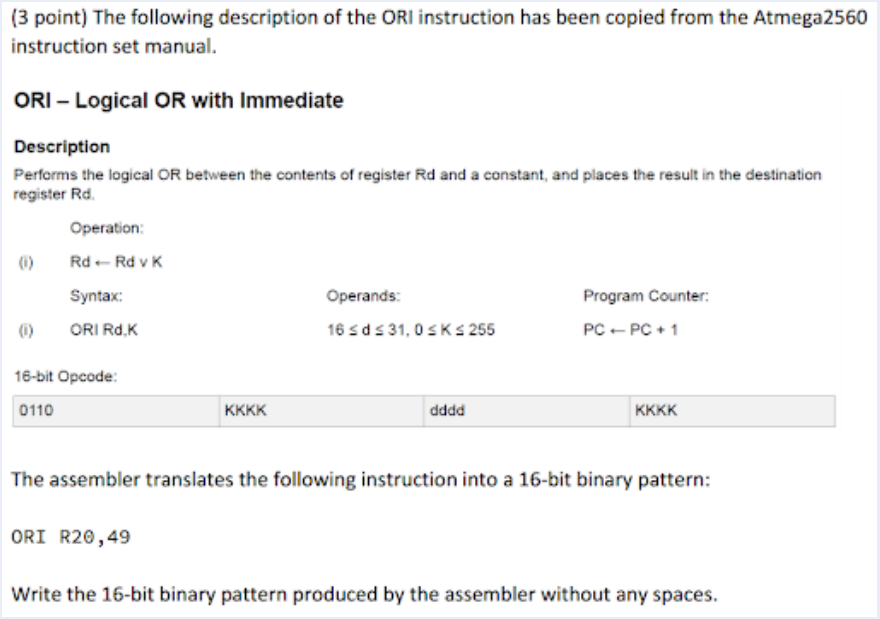
Converting ORI Instruction to Binary

01/04/2026 19:33:43

Based on the instruction manual provided in the image, here is the step-...



(3 point) The following description of the ORI instruction has been copied from the Atmega2560 instruction set manual.

ORI – Logical OR with Immediate

Description

Performs the logical OR between the contents of register Rd and a constant, and places the result in the destination register Rd.

Operation:

(i) Rd ← Rd v K

	Syntax:	Operands:	Program Counter:
(i)	ORI Rd,K	16 ≤ d ≤ 31, 0 ≤ K ≤ 255	PC ← PC + 1

16-bit Opcode:

0110	KKKK	dddd	KKKK

The assembler translates the following instruction into a 16-bit binary pattern:

```
ORI R20,49
```

Write the 16-bit binary pattern produced by the assembler without any spaces.

Based on the instruction manual provided in the image, here is the step-by-step breakdown to convert `ORI R20, 49` into its 16-bit binary pattern.

1. Analyze the Instruction Format

From the image, the 16-bit opcode structure is: `0110 KKKK dddd KKKK`

Where:

- **0110**: The fixed operation code.
- **K**: Represents the 8-bit immediate constant (49). The bits are split into upper (`KKKK`) and lower (`KKKK`) nibbles.
- **d**: Represents the destination register (R20). The manual states `16 ≤ d ≤ 31`. The 4-bit field `dddd` represents the register number minus the offset of 16.

2. Convert the Operands

Constant K (49):

- Convert decimal 49 to 8-bit binary:

$$49 = 32 + 16 + 1 = 00110001_2$$

- Split into upper and lower 4-bit segments (nibbles):
  - **Upper K (bits 11-8):** `0011`
  - **Lower K (bits 3-0):** `0001`

Register Rd (R20):

- The instruction works on registers R16 to R31.
- Subtract the offset (16) from the register number:

$$20 - 16 = 4$$

- Convert 4 to a 4-bit binary value:
  - **d (bits 7-4):** `0100`

3. Assemble the Binary Pattern

Now, place the calculated bits into the opcode template: `0110` `Upper K` `d` `Lower K`

Section	Fixed Opcode	Upper K (from 49)	d (from R20)	Lower K (from 49)
Bits	`0110`	`0011`	`0100`	`0001`

Final Answer

Combining these parts creates the 16-bit binary pattern:

0110001101000001