****

1. **Instruction: addi sp, sp, -16 (ff010113)**

* **Opcode:** 0010011 (7 bits)
* **Immediate:** -16 (12 bits, two's complement)
* **Source Register (rs1):** sp (x2, 5 bits)
* **Destination Register (rd):** sp (x2, 5 bits)
* **Function (funct3):** 000 (3 bits)

**Breakdown:**

* **Immediate (-16):** 111111110000
* **rs1 (sp = x2):** 00010
* **funct3:** 000
* **rd (sp = x2):** 00010
* **Opcode:** 0010011

imm[11:0] | rs1 | funct3 | rd | opcode

111111110000 | 00010 | 000 | 00010 | 0010011



1. **Instruction: mv a1,a0 (00050593)**

* **Opcode:** 0010011 (7 bits)
* **Immediate:** 0 (12 bits, two's complement)
* **Source Register (rs1):** a0 (x10, 5 bits)
* **Destination Register (rd):** a1 (x11, 5 bits)
* **Function (funct3):** 000 (3 bits)

**Breakdown:**

* **Immediate (0):** 000000000000
* **rs1 (a0 = x10):** 01010
* **funct3:** 000
* **rd (a1 = x11):** 01011
* **Opcode:** 0010011

imm[11:0] | rs1 | funct3 | rd | opcode

000000000000 | 01010 | 000 | 01011 | 0010011



**3. Instruction: sd s0, 0(sp) (00813023)**

* **Opcode:** 0100011 (7 bits)
* **Immediate:** 0 (12 bits, split into two parts: imm[11:5] and imm[4:0])
* **Source Register (rs2):** s0 (x8, 5 bits)
* **Base Register (rs1):** sp (x2, 5 bits)
* **Function (funct3):** 011 (3 bits)

**Breakdown:**

* **Immediate (112):** 000000000000 (split into imm[11:5] = 0000000 and imm[4:0] = 10000)
* **rs2 (s0 = x8):** 01000
* **rs1 (sp = x2):** 00010
* **funct3:** 011
* **Opcode:** 0100011

**Binary Representation:**

* imm[11:5] (7 bits): 0000000
* rs2 (5 bits): 01000
* rs1 (5 bits): 00010
* funct3 (3 bits): 011
* imm[4:0] (5 bits): 00000
* Opcode (7 bits): 0100011



**4. Instruction: lui a0, 0x21 (00021537)**

* **Opcode**: 0110111 (7 bits for the lui instruction)
* **Immediate**: 0x21 (20 bits, upper 20 bits of the 32-bit immediate)
* **Destination Register (rd)**: a0 (x10, 01010 in 5 bits)

**Field Breakdown:**

1. **Immediate (imm[31:12])**:  
   The upper 20 bits of the immediate value 0x21 are:  
   0x21 = 00000000000100001 in binary (20 bits):  
   **00000000000100001**
2. **Destination Register (rd)**:  
   a0=x10 , represented in **5-bit binary**:  
   **01010**
3. **Opcode**:  
   lui instruction, opcode=0110111  
   **0110111**

**Binary Representation:**

imm[31:12] | rd | opcode

00000000000100001 | 01010 | 0110111

**Full binary: 00000000000100001010 0110111**

**Hex: 00021537**



**5.Instruction: sw a3, 172(a4) (0ad72623)**

* **Opcode**: 0100011 (7 bits for the sw instruction)
* **Immediate**: 172 (split into 12 bits, in two parts: 7 bits and 5 bits)
* **Source Register 1 (rs1)**: a4 (x14, 01100 in 5 bits)
* **Source Register 2 (rs2)**: a3 (x13, 01011 in 5 bits)
* **Function (funct3)**: 010 (for store word, sw)

**Field Breakdown:**

* **Immediate**:
* **Immediate [11:5]** (7 bits):  
  172172 in binary = 10101100, so the higher 7 bits are:  
  **1010110** (7 bits).
* **Immediate [4:0]** (5 bits):  
  The lower 5 bits are:  
  **11000**.
* **Source Register 1 (rs1)**:  
  a4=x14, represented in **5-bit binary**:  
  **01100**
* **Source Register 2 (rs2)**:  
  a3=x13, represented in **5-bit binary**:  
  **01011**
* **Function (funct3)**:  
  The function for store word (sw) is:  
  **010**
* **Opcode**:  
  sw instruction, opcode=0100011:  
  **0100011**

**Binary Representation:**

imm[11:5] | rs2 | rs1 | funct3 | imm[4:0] | opcode

1010110 | 01011 | 01100 | 010 | 11000 | 0100011

Full binary: **1010110 01011 01100 010 11000 0100011**

**Hex: 0ad72623**

****

**6.Instruction: ld a5, 0(s0) (00043783)**

* **Opcode**: 0000011 (7 bits for the ld instruction)
* **Immediate**: 0 (12 bits in 2's complement, 000000000000)
* **Source Register (rs1)**: s0 (x8, 01000 in 5 bits)
* **Destination Register (rd)**: a5 (x15, 01111 in 5 bits)
* **Function (funct3)**: 011 (for load double-word, ld)

**Field Breakdown:**

* **Immediate (imm[11:0])**:  
  Immediate value is 0, represented in **12-bit binary**:  
  **000000000000**
* **Source Register (rs1)**:  
  s0=x8, represented in **5-bit binary**:  
  **01000**
* **Function (funct3)**:  
  For ld, the function is 011:  
  **011**
* **Destination Register (rd)**:  
  a5=x15 , represented in **5-bit binary**:  
  **01111**
* **Opcode**:  
  ld instruction, opcode = 0000011:  
  **0000011**

**Binary Representation:**

imm[11:0] | rs1 | funct3 | rd | opcode

000000000000 | 01000 | 011 | 01111 | 0000011

Full binary: **000000000000 01000 011 01111 0000000 0000011**

**Hex: 00043783**

**7. Instruction: jal ra, 10414 <printf>**

* **Opcode**: 1101111 (7 bits for the jal instruction)
* **Immediate**: 10414 (20 bits in hexadecimal, 0x10414)
* **Destination Register (rd)**: ra (x1, 00001 in 5 bits)

**Immediate Breakdown:**

* imm[20]: 0
* imm[10:1]: 0000000100
* imm[11]: 0
* imm[19:12]: 00010001

**Field Breakdown:**

* **Immediate**:
  + imm[20|10:1|11|19:12]:  
    **0 | 0000000100 | 0 | 00010001**
  + Combine them into a 20-bit immediate:  
    **000000010000000000010001** (20 bits).
* **Destination Register (rd)**:  
  ra=x1, represented in **5-bit binary**:  
  **00001**
* **Opcode**:  
  jal instruction, opcode = 1101111:  
  **1101111**

**Binary Representation:**

imm[20|10:1|11|19:12] | rd | opcode

000000010000000000010001 | 00001 | 1101111

Full binary: **000000010000000000010001 00001 1101111**

**Hex: 26c000ef**

****

**8. Instruction: li a1, 50**

* **Opcode**: 0010011 (7 bits for the li instruction)
* **Immediate**: 50 (12 bits in decimal)
* **Source Register (rs1)**: x0 (no register, so 00000 in 5 bits)
* **Destination Register (rd)**: a1 (x11, 01011 in 5 bits)
* **Function (funct3)**: 000 (for the li instruction)

**Field Breakdown:**

* **Immediate (imm[11:0])**:  
  The immediate value is 50, represented in **12-bit binary**:  
  **000000000110**
* **Source Register (rs1)**:  
  Since li uses the addi instruction with x0 as the source, rs1 = x0, which is 00000 in 5 bits.  
  **00000**
* **Function (funct3)**:  
  The function code for addi is 000:  
  **000**
* **Destination Register (rd)**:  
  a1=x11, represented in **5-bit binary**:  
  **01011**
* **Opcode**:  
  The opcode for li is 0010011:  
  **0010011**

**Binary Representation:**

imm[11:0] | rs1 | funct3 | rd | opcode

000000000110 | 00000 | 000 | 01011 | 0010011

Full binary: **000000000110 00000 000 01011 0010011**

**Hex: 03200593**

****

**9.Instruction: bnez a5, 10190 <main+0xc>**

1. **Opcode**: 1100011 (7 bits for branch instructions)
2. **funct3**: for BNE (Branch if Not Equal), funct3 is 001
3. **Immediate**: 10190 (decimal value for the immediate)
4. **Source Register 1 (rs1)**: a5 (x15, 01111 in 5 bits)
5. **Source Register 2 (rs2)**: x0 (since we are using bnez, rs2 is x0, which is 00000 in 5 bits)

**Immediate Breakdown:**

* imm[12]: 0
* imm[10:5]: 010000
* imm[4:1]: 1111
* imm[11]: 0

**Field Breakdown:**

* **Immediate**: imm[12|10:5|4:1|11]:  
  **0 | 010000 | 1111 | 0**
* **Source Register 1 (rs1)**:  
  a5=x15, represented in **5-bit binary**:  
  **01111**
* **Source Register 2 (rs2)**:  
  Since bnez, rs2 = x0, represented in **5-bit binary**:  
  **00000**
* **Function (funct3)**:  
  The function code for BNE is 001:  
  **001**
* **Opcode**:  
  The opcode for branch operations is 1100011:  
  **1100011**

**Binary Representation:**

imm[12|10:5|4:1|11] | rs2 | rs1 | funct3 | imm[4:1] | opcode

001000 | 00000 | 01111 | 001 | 1111 | 1100011

Full binary: **0001000000001111001000110001100011**

**Hex: FE079EE3**

****

**10. Instruction: beqz a5, 101fc <exit+0x2c>**

* **Opcode**: 1100011 (7 bits for branch instructions)
* **funct3**: for BEQZ (Branch if Equal to Zero), funct3 is 000 .
* **Immediate**: 101fc (decimal value for the immediate)
* **Source Register 1 (rs1)**: a5 (x15, 01111 in 5 bits)
* **Source Register 2 (rs2)**: x0 (since we are using beqz, rs2 is x0, which is 00000 in 5 bits)

**Immediate Breakdown:**

* imm[12]: 1
* imm[10:5]: 111111
* imm[4:1]: 0000
* imm[11]: 1

**Field Breakdown:**

* **Immediate**:

imm[12|10:5|4:1|11]:  
**1 | 111111 | 0000 | 1**

* **Source Register 1 (rs1)**:  
  a5=x15, represented in **5-bit binary**:  
  **01111**
* **Source Register 2 (rs2)**:  
  Since beqz, rs2 = x0, represented in **5-bit binary**:  
  **00000**
* **Function (funct3)**:  
  The function code for BEQ and BEQZ is 000:  
  **000**
* **Opcode**:  
  The opcode for branch operations is 1100011:  
  **1100011**

**Binary Representation:**

imm[12|10:5|4:1|11] | rs2 | rs1 | funct3 | imm[4:1] | opcode

111111 | 00000 | 01111 | 000 | 0000 | 1100011

Full binary: **11111100000100000000011110000011**

**Hex: 00078463**

****

**11.Instruction: auipc a5, 0xffff0**

* **Opcode:** 0010111 (7 bits for AUIPC)
* **Immediate:** 0xffff0 (20 bits in hexadecimal)
* **Destination Register (rd):** a5 (x15, 01111 in 5 bits)

**Immediate Breakdown:**

* imm[31:12]: 111111111111
* imm[11:0]: 000000000000

**Field Breakdown:**

1. **Immediate (imm[31:12]):**  
   111111111111 (12 bits)
2. **Destination Register (rd):**  
   a5 = x15, represented in 5-bit binary:  
   01111
3. **Opcode:**  
   The opcode for AUIPC is 0010111:  
   0010111

**Binary Representation:**

imm[31:12] | rd | opcode

111111111111 | 01111 | 0010111

**Full binary:** 11111111111100000000 01111 0010111

**Hex: ffff0797**

****

**12.Instruction: addiw a5, a5, -1**

* **Opcode**: 0010011 (7 bits for the ADDIW instruction )
* **Immediate**: -1 (12 bits in two's complement)
* **Source Register (rs1)**: a5 (x15, 01111 in 5 bits)
* **Destination Register (rd)**: a5 (x15, 01111 in 5 bits)
* **Function (funct3)**: 001 for ADDIW

**Breakdown:**

* **Immediate (-1)**:
* -1 in two's complement = 111111111111 (12 bits)
* **Source Register (rs1)**:
* a5 = x15, represented in **5-bit binary**:  
  **01111**
* **Destination Register (rd)**:
* a5 = x15, represented in **5-bit binary**:  
  **01111**
* **Function (funct3)**:
* ADDIW has funct3 as 001:  
  **001**
* **Opcode**:
* The opcode for ADDIW is 0010011:  
  **0010011**

**Binary Representation:**

imm[11:0] | rs1 | funct3 | rd | opcode

111111111111 | 01111 | 001 | 01111 | 0010011

Full binary: **11111111111101111001011110010011**

Hex: fff7879b



**13. Instruction: sub a5, a5, s0**

* **Opcode**: 0110011 (7 bits) – This opcode is for **integer register-register operations**.
* **Funct3**: 000 (3 bits)
* **Funct7**: 0100000 (7 bits) – This specifies the **subtraction** operation (using the sub instruction).
* **rs1**: a5 (x15, 5 bits) – The first source register is a5, which is x15
* **rs2**: s0 (x8, 5 bits) – The second source register is s0, which is x8.
* **rd**: a5 (x15, 5 bits) – The destination register is a5, which is x15.

**Breakdown:**

* **Opcode**: 0110011 (for register-register operations)
* **rd (Destination Register)**: a5 = x15, represented in **5-bit binary**:  
  **01111**
* **funct3**: 000
* **rs1 (First Source Register)**: a5 = x15, represented in **5-bit binary**:  
  **01111**
* **rs2 (Second Source Register)**: s0 = x8, represented in **5-bit binary**:  
  **01000**
* **funct7**: 0100000 (7 bits) – This specifies the sub operation.

**Binary Representation:**

0100000 | 01000 | 01111 | 000 | 01111 | 0110011

Full binary: **0100000 01000 01111 000 01111 0110011**

Hex: 408787b3

**14. Instruction: add a2, a2, a5**

* **Opcode**: 0110011 (7 bits) – This opcode is for **integer register-register operations**.
* **Funct3**: 000 (3 bits)
* **Funct7**: 0000000 (7 bits) – This specifies the **add** operation.
* **rs1**: a2 (x12, 5 bits) – The first source register is a2, which is x12 in RISC-V.
* **rs2**: a5 (x15, 5 bits) – The second source register is a5, which is x15 in RISC-V.
* **rd**: a2 (x12, 5 bits) – The destination register is a2, which is x12.

**Breakdown:**

* **Opcode**: 0110011 (for register-register operations)
* **rd (Destination Register)**: a2 = x12, represented in **5-bit binary**:  
  **01100**
* **funct3**: 000
* **rs1 (First Source Register)**: a2 = x12, represented in **5-bit binary**:  
  **01100**
* **rs2 (Second Source Register)**: a5 = x15, represented in **5-bit binary**:  
  **01111**
* **funct7**: 0000000 (7 bits) – This specifies the add operation.

**Binary Representation:**

0000000 | 01111 | 01100 | 000 | 01100 | 0110011

Full binary: **0000000 01111 01100 000 01100 0110011**

Hex: 00f60633



**15**. **Instruction: jal ra, 18b50 <strlen>**

* **Opcode**: 1101111 (7 bits) – This opcode is for the **JAL** (Jump and Link) instruction.
* **Immediate (imm)**: 0x18b50 (20 bits) – This represents the offset for the jump, relative to the program counter (PC).
* **Destination Register (rd)**: ra (x1, 5 bits) – The destination register is ra, which is x1.

**Immediate (Offset): 0x18b50**

**Binary (20 bits):** 00011000101101010000  
imm[20|10:1|11|19:12]: 0|0101101010|1|00011000

**Instruction Fields:**

* **rd**: ra = x1 – Destination register is x1, which is 00001 in binary.
* **opcode**: 1101111 – Opcode for jal.

**Binary Representation:**

imm[20|10:1|11|19:12] | rd | opcode

0|0101101010|1|00011000 | 00001 | 1101111

Binary: 0000 1011 0101 0000 1100 0001 1101 1111

Hex: 68c080ef