RV32I Control Timing Diagrams

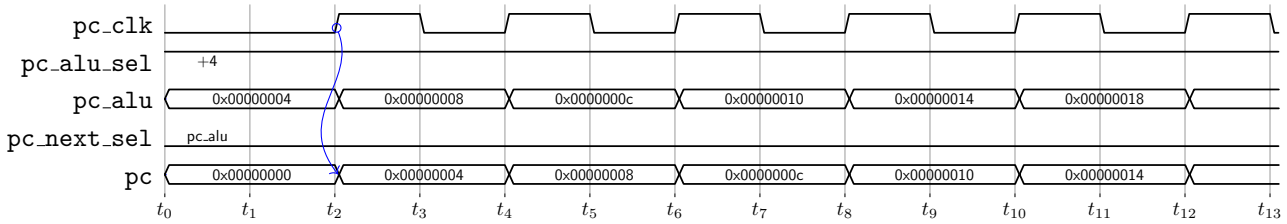

Figure 1: Program counter behavior.

```
00000000: 000087b7  lui     x15,0x8       // x15 = 0x00008000
00000004: 12345117  auipc   x2,0x12345    // x2 = 0x00000004 + 0x12345000 = 0x12345004
00000008: 00a10093  addi    x1,x2,10      // x1 = 0x12345004 + 0x0000000a = 0x1234500e
0000000c: 4027da13  srai    x20,x15,2     // x20 = 0x00008000 >> 2 = 0x00002000
00000010: 00f100b3  add     x1,x2,x15     // x1 = 0x12345004 + 0x00008000 = 0x1234d004
00000014: 402a00b3  sub     x1,x20,x2     // x1 = 0x00002000 - 0x12345004 = 0xedcbcffc
```

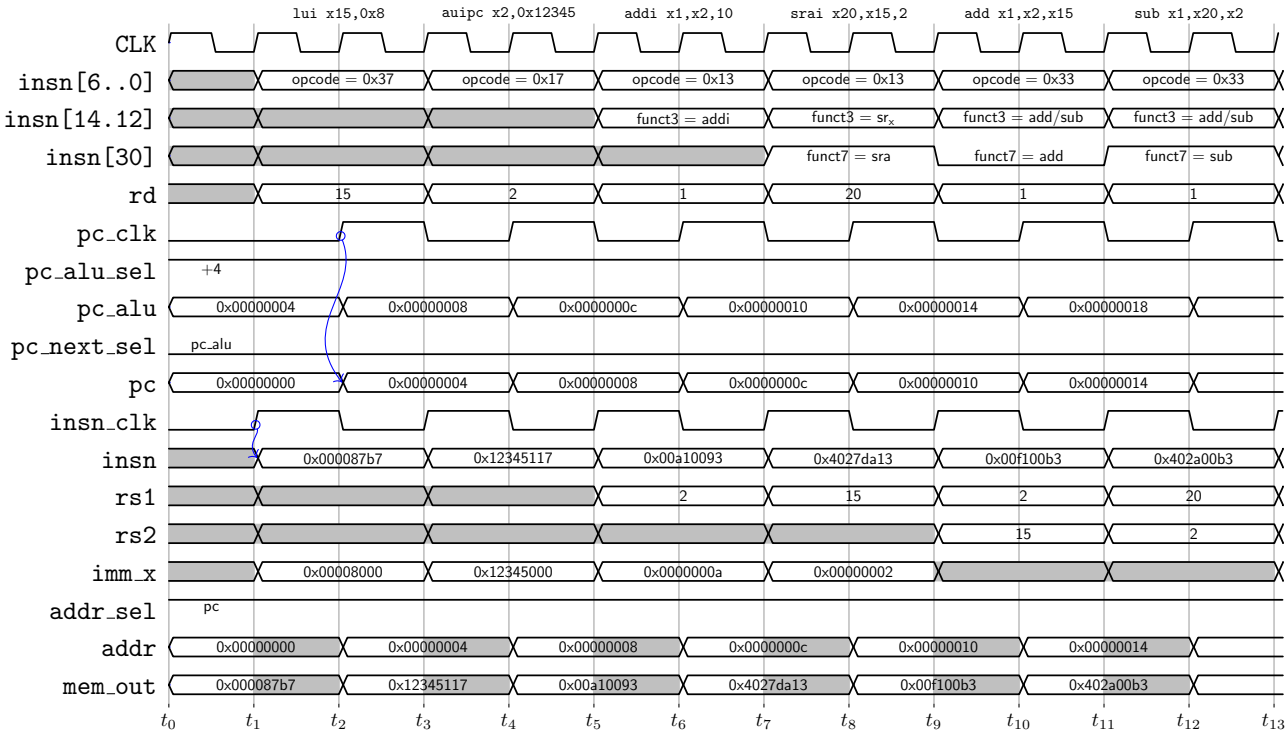

Figure 2: Fetch and decode cycles of U-type, R-type, and some I-type instructions.

```
00000000: 000087b7  lui     x15,0x8      // x15 = 0x00008000
00000004: 12345117  auipc   x2,0x12345   // x2 = 0x00000004 + 0x12345000 = 0x12345004
00000008: 00a10093  addi    x1,x2,10     // x1 = 0x12345004 + 0x0000000a = 0x1234500e
0000000c: 4027da13  srai    x20,x15,2    // x20 = 0x00008000 >> 2 = 0x00002000
00000010: 00f100b3  add     x1,x2,x15    // x1 = 0x12345004 + 0x00008000 = 0x1234d004
00000014: 402a00b3  sub     x1,x20,x2    // x1 = 0x00002000 - 0x12345004 = 0xedcbcffc
```

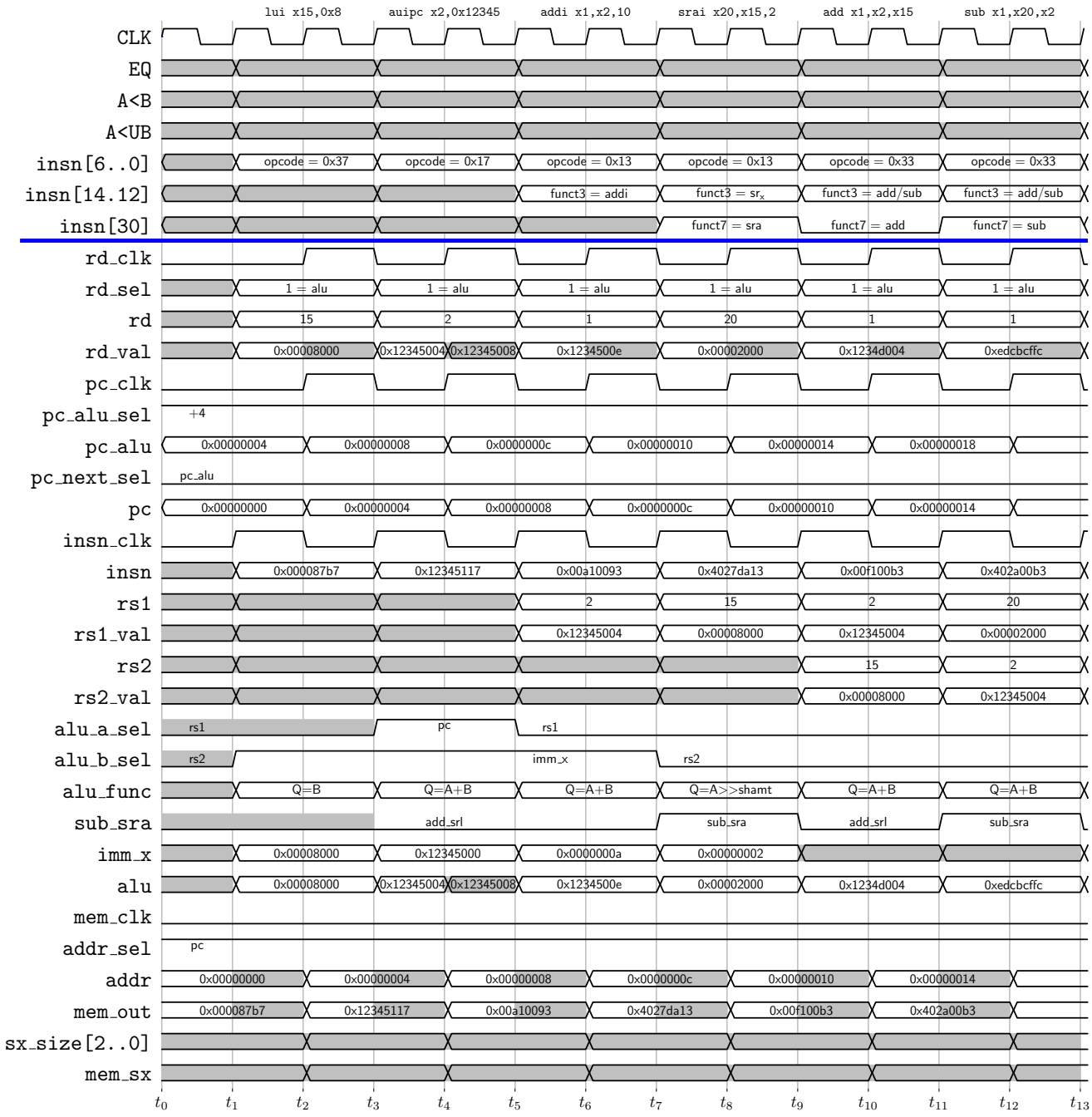

Figure 3: Fetch, decode, and execute cycles of U-type, R-type, and some I-type instructions.

```
00000018: 0027a423  sw      x2,8(x15)    // m32(0x00008000 + 0x00000008) = 0x12345004
0000001c: 0087a803  lw      x16,8(x15)   // x16 = sx(m32(0x00008000 + 0x00000008)) = 0x12345004
00000020: 008000ef  jal     x1,0x28      // x1 = 0x00000024,  pc = 0x00000020 + 0x00000008 = 0x00000028

00000028: 03000267  jalr    x4,48(x0)    // x4 = 0x0000002c,  pc = (0x00000030 + 0x00000000) & 0xfffffffe = 0x00000030

00000030: 02f00063  beq     x0,x15,0x50  // pc += (0x00000000 == 0x00008000 ? 0x00000020 : 4) = 0x00000034
00000034: 03010063  beq     x2,x16,0x54  // pc += (0x12345004 == 0x12345004 ? 0x00000020 : 4) = 0x00000054
```

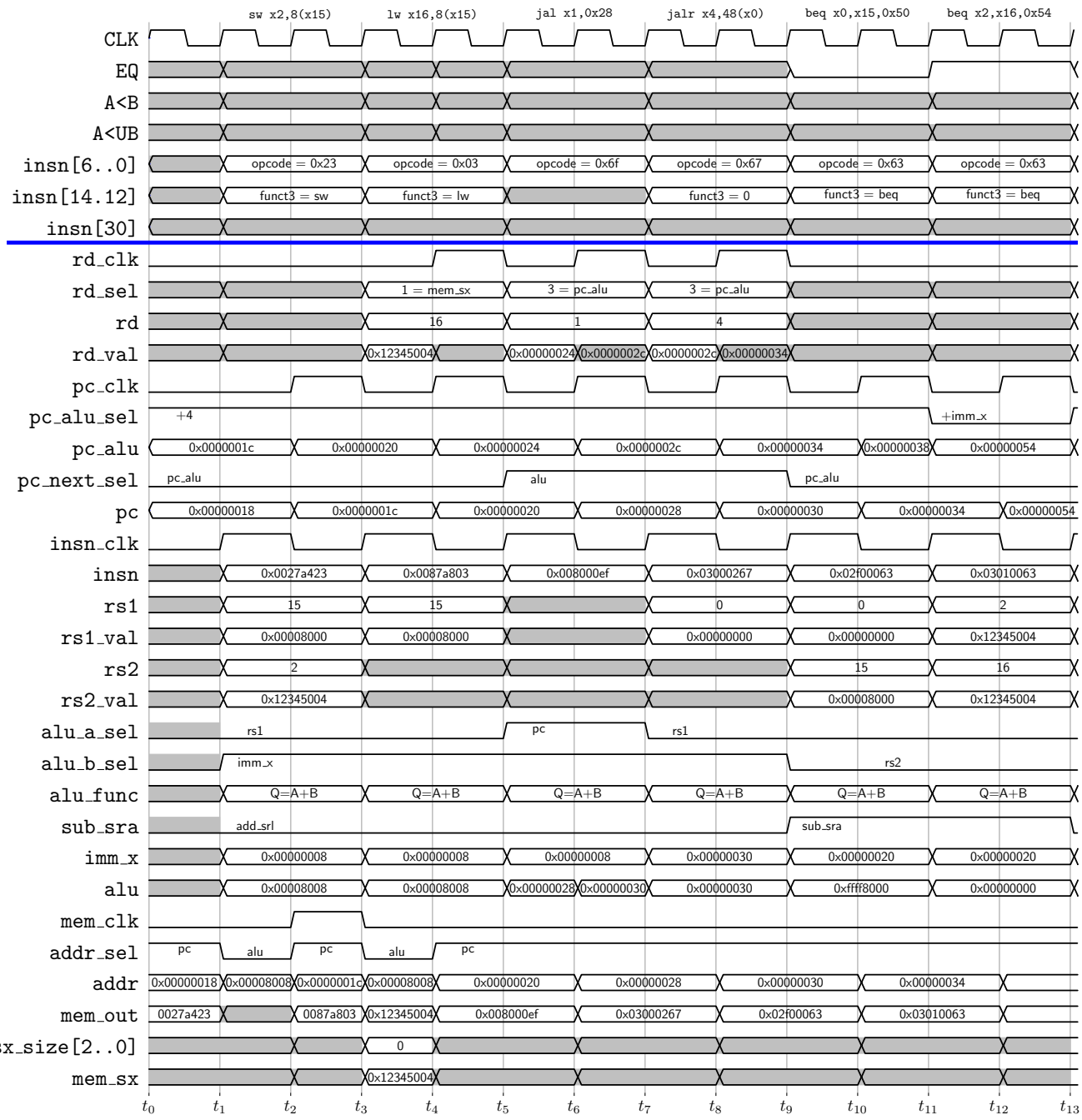

Figure 4: Fetch, decode, and execute of load, store, jump, and branch instructions.