ZX16 Binary Test Files

TC-ZX16-01.bin (Hello World)

```
37010100 13053000 B7050010 73000000 1305A000 73000000
```

String data at offset 0x1000:

```
48656C6C6F2C205A58313620576F726C6421000000000000
```

TC-ZX16-02.bin (Input/Output Test)

```
13051000 73000000 13053000 73000000 13052000 73000000 1305A000 73000000
```

TC-ZX16-03.bin (Register Operations)

```
93004006 1301800C B3012000 33821040 13058000 73000000 1305A000 73000000
```

TC-ZX16-04.bin (Memory Access)

```
37200020 13014023 23202000 83210000 13059000 93050000 13060001 73000000 1305A000 73000000
```

TC-ZX16-05.bin (Branch/Loop Test)

```
9300A000 13010000 63081100 93011100 EFF09FFF 13058000 73000000 1305A000 73000000
```

TC-ZX16-06.bin (Graphics Memory Test)

```
37F00F00 13011004 23002000 23082000 13059000 93050000 13060002 73000000 1305A000 73000000
```

TC-ZX16-07.bin (Fibonacci Generator)

```
93000000 13011000 9301A000 13020000 63083201 B3022000 93002000 13015000 13021200 EFF01FFE
13058000 73000000 1305A000 73000000
```

Creating Binary Files

To create these files for your simulator, you'll need to convert the hex strings to actual binary data. Here are the commands for each test case:

Linux/Mac Terminal:

bash

```
# TC-ZX16-01.bin
echo "37010100 13053000 B7050010 73000000 1305A000 73000000" | xxd -r -p > TC-ZX16-01.bin

# TC-ZX16-02.bin
echo "13051000 73000000 13053000 73000000 13052000 73000000 1305A000 73000000" | xxd -r -p > TC

# TC-ZX16-03.bin
echo "93004006 1301800C B3012000 33821040 13058000 73000000 1305A000 73000000" | xxd -r -p > TC

# TC-ZX16-04.bin
echo "37200020 13014023 23202000 83210000 13059000 93050000 13060001 73000000 1305A000 7300000

# TC-ZX16-05.bin
echo "9300A000 13010000 63081100 93011100 EFF09FFF 13058000 73000000 1305A000 73000000" | xxd -

# TC-ZX16-06.bin
echo "37F00F00 13011004 23002000 23082000 13059000 93050000 13060002 73000000 1305A000 7300000

# TC-ZX16-07.bin
echo "93000000 13011000 9301A000 13020000 63083201 B3022000 93002000 13015000 13021200 EFF01FFE
```

Python Script to Generate Files:

```python
import binascii

test_cases = {
    "TC-ZX16-01.bin": "37010100 13053000 B7050010 73000000 1305A000 73000000",
    "TC-ZX16-02.bin": "13051000 73000000 13053000 73000000 13052000 73000000 1305A000 73000000"
    "TC-ZX16-03.bin": "93004006 1301800C B3012000 33821040 13058000 73000000 1305A000 73000000"
    "TC-ZX16-04.bin": "37200020 13014023 23202000 83210000 13059000 93050000 13060001 73000000
    "TC-ZX16-05.bin": "9300A000 13010000 63081100 93011100 EFF09FFF 13058000 73000000 1305A000
    "TC-ZX16-06.bin": "37F00F00 13011004 23002000 23082000 13059000 93050000 13060002 73000000
    "TC-ZX16-07.bin": "93000000 13011000 9301A000 13020000 63083201 B3022000 93002000 13015000
}

for filename, hex_data in test_cases.items():
    # Remove spaces and convert to binary
    hex_string = hex_data.replace(" ", "")
    binary_data = binascii.unhexlify(hex_string)

    with open(filename, 'wb') as f:
        f.write(binary_data)

    print(f"Created {filename} ({len(binary_data)} bytes)")
```

Expected Behavior for Each Test:

TC-ZX16-01.bin

- Should load immediate values and call string output service
- Expected output: "Hello, ZX16 World!" (if string is loaded at 0x1000)

TC-ZX16-02.bin

- Tests input/output services
- Should prompt for string input, echo it, then prompt for integer

TC-ZX16-03.bin

- Tests basic arithmetic operations
- Should show register dump with x1=100, x2=200, x3=300, x4=100

TC-ZX16-04.bin

- Tests memory store/load operations
- Should show memory dump at 0x2000 with stored value 0x1234

TC-ZX16-05.bin

- Tests branch and jump instructions
- Should count from 0 to 10 in a loop

TC-ZX16-06.bin

- Tests graphics memory access
- Should write to graphics tile map area at 0xF000

TC-ZX16-07.bin

- Complex program testing multiple features
- Generates Fibonacci sequence numbers

File Sizes:

- TC-ZX16-01.bin: 24 bytes
- TC-ZX16-02.bin: 32 bytes
- TC-ZX16-03.bin: 32 bytes
- TC-ZX16-04.bin: 40 bytes
- TC-ZX16-05.bin: 36 bytes
- TC-ZX16-06.bin: 40 bytes
- TC-ZX16-07.bin: 56 bytes

These binary files contain the actual machine code instructions that your ZX16 simulator needs to execute. Each 32-bit instruction is stored in little-endian format as required by the RISC-V specification.