Lab2_2 & Exam1_review

Yuanqing Miao

Introduction to UTF-8 Encoding

- UTF-8 is a variable-width character encoding used for electronic communication.

It represents characters using 1 to 4 bytes:

- 1 byte: ASCII characters (0x00 - 0x7F)

- 2 bytes: Extended Latin, Greek, Cyrillic, etc.

- 3 bytes: Most common Unicode characters

- 4 bytes: Rare characters, including emojis

UTF-8 Encoding Rules

- Character number range (hex) -> UTF-8 encoding:

- - 0x0000 - 0x007F -> 0xxxxxxx (1 byte)
- - 0x0080 - 0x07FF -> 110xxxxx 10xxxxxx (2 bytes)
- - 0x0800 - 0xFFFF -> 1110xxxx 10xxxxxx 10xxxxxx (3 bytes)
- - 0x10000 - 0x10FFFF -> 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx (4 bytes)

MIPS Assembly Implementation

- 1. Load the Unicode code point from memory
- 2. Check the range using unsigned comparisons
- 3. Encode the value based on its size:
-     - 1-byte (ASCII)
-     - 2-byte encoding
-     - 3-byte encoding
-     - 4-byte encoding
- 4. Store and print the UTF-8 encoded value

Example: Encoding U+0800 (0x0800) to UTF-8

Step 1: Determine Encoding Size

- - U+0800 falls in the range `0x0800 - 0xFFFF` → 3-byte encoding

Step 2: Binary Representation of Code Point

- - U+0800 (hex) = `0000 1000 0000 0000` (binary)

Step 3: UTF-8 Encoding (3 Bytes)

First byte: `1110xxxx`

- Extract top 4 bits: `0000` → `11100000` (`0xE0`)

Second byte: `10xxxxxx`

- Next 6 bits: `001000` → `10100000` (`0xA0`)

Third byte: `10xxxxxx`

- Last 6 bits: `000000` → `10000000` (`0x80`)

Final UTF-8 Encoding:

- Binary: `11100000 10100000 10000000`
- Hexadecimal: `0xE0 A0 80`

MIPS Assembly: 3-Byte Encoding

```
encode_3_bytes:
    # UTF-8: 1110xxxx 10xxxxxx 10xxxxxx
    li $t3, 0xE0       # First byte base: 1110xxxx
    srl $t4, $t1, 12   # Get upper 4 bits
    andi $t4, $t4, 0x0F
    or $t3, $t3, $t4   # Merge

    li $t5, 0x80       # Second byte base: 10xxxxxx
    srl $t6, $t1, 6
    andi $t6, $t6, 0x3F
    or $t5, $t5, $t6   # Merge

    li $t7, 0x80       # Third byte base: 10xxxxxx
    andi $t8, $t1, 0x3F
    or $t7, $t7, $t8   # Merge
```

Exam1_review

B. For the MIPS R-format instructions, what are the field names, and how many bits do they contain?

Field Name	Bit Width	Description
opcode	6 bits	Specifies the operation type (for R-format, this is always `000000`).
rs	5 bits	Source register 1.
rt	5 bits	Source register 2.
rd	5 bits	Destination register.
shamt	5 bits	Shift amount (used in shift instructions, otherwise `00000`).
funct	6 bits	Function code specifying the exact operation (e.g., `100000` for `add`).

Total: **32 bits** (6 + 5 + 5 + 5 + 5 + 6)

D. Provide the type, assembly language instruction, and binary representation of instruction described by the following MIPS fields: (5 points)

op = 0, rs = 3, rt = 2, rd = 3, shamt = 0, funct = 34

Field Breakdown:

Field	Value	Description
op	000000	R-type instruction (opcode = 0)
rs	00011	Source register 1 (`$3`)
rt	00010	Source register 2 (`$2`)
rd	00011	Destination register (`$3`)
shamt	00000	Shift amount (not used)
funct	100010	Function code for `sub`

Binary Representation:

```
000000 00011 00010 00011 00000 100010
```

Hex Representation:

```
0x00621822
```

E. Computer A has an overall CPI of 1.3 and can be run at a clock rate of 600MHz. Computer B has a CPI of 2.5 and can be run at a clock rate of 750 Mhz. We have a particular program we wish to run. When compiled for computer A, this program has exactly 100,000 instructions. How many instructions would the program need to have when compiled for Computer B, in order for the two computers to have exactly the same execution time for this program? (6 points)

Given Data:

- **Computer A:**
  - **CPI:** 1.3
  - **Clock Rate:** 600 MHz
  - **Instruction Count:** 100,000
- **Computer B:**
  - **CPI:** 2.5
  - **Clock Rate:** 750 MHz
  - **Unknown Instruction Count:** ?

Execution Time Formula:

$$\text{Execution Time} = \frac{\text{Instruction Count} \times \text{CPI}}{\text{Clock Rate}}$$

For **Computer A:**

$$T_A = \frac{100,000 \times 1.3}{600 \times 10^6}$$

$$T_A = \frac{130,000}{600 \times 10^6} = 2.1667 \times 10^{-4} \, \text{seconds}$$

For **Computer B:** (Let `X` be the instruction count)

$$T_B = \frac{X \times 2.5}{750 \times 10^6}$$

Setting **T_A = T_B**:

$$2.1667 \times 10^{-4} = \frac{X \times 2.5}{750 \times 10^6}$$

Solving for `X`:

$$X = \frac{(2.1667 \times 10^{-4}) \times (750 \times 10^6)}{2.5}$$

$$X = 65,000 \text{ instructions}$$

F. Assume the following register contents:

$t0 = 0xAAAAAAAA, $t1 = 0x12345678

ii. For the register values shown above, what is the value of $t2 for the following sequence of instructions? (4 points)

```
sll $t2, $t0, 4
or  $t2, $t2, $t1
```

1. `sll $t2, $t0, 4` (Shift `$t0` left by 4 bits)
   - Binary Representation of `$t0`:

     `1010 1010 1010 1010 1010 1010 1010 1010`
   - After shifting left by 4 bits:

     `1010 1010 1010 1010 1010 1010 1010 0000`
   - Result: `0xAAAAAAA0`
   - **$t2** = `0xAAAAAAA0`

2. `or $t2, $t2, $t1` (Bitwise OR with `$t1`)

    - Binary Representation:

```
0xAAAAAAA0 = 1010 1010 1010 1010 1010 1010 1010 0000
0x12345678 = 0001 0010 0011 0100 0101 0110 0111 1000
-----------------------------------------------------
OR result  = 1011 1010 1011 1110 1111 1110 1111 1000
```

    - Result: `0xBABEFEF8`
    - **Final Value of** `$t2` = `0xBABEFEF8`