1 RISC-y Conversions

1.1 Convert the different RISC-V commands into their hex form or convert the hex form into RISC-V. The instructions are in order of when they would be executed.

(a) `0x005004B3`

(b) `lw t5, 17(t6)`

(c) `sll s9, x9, t0`

(d) `0x03CE2283`

(e) `jalr a0, x11, 8`

(f) `ori t1, t2, 5`

(g) `lui a7, 0xCF61C`

2 Linked List Reversals in RISC-V

2.1 Assume we have the following linked list node struct:

```
struct node{
    int val;
    struct node * next;
};
```

Also, recall the function to reverse a linked list iteratively, given a pointer to the head of the linked list.

```
void reverse(struct node * head){
    struct node * prev = NULL;
    struct node * next;
    struct node * curr = head;
    while(curr != NULL){
        next = curr->next;
        curr->next = prev;
        prev = curr;
        curr = next;
    }
}
```

2.2 Now assume `a0` contains the address of the head of a linked list. Fill in the function below to reverse a linked list. Assume reverse follows calling conventions. reverse doesnt return anything. You may not need all lines.

```
1.  reverse: _____
2.      _____
3.      _____
4.      _____
5.  add s0 a0 x0
6.  xor s2 s2 s2 #s2 corresponds to the pointer prev
7.  loop: ___ s0 x0 exit
8.      _____
9.      _____
10. add s2 s0 x0
11. add s0 s1 x0
12. j loop
13. exit: _____
14.     _____
15.     _____
16. addi sp sp 12
17. j ra
```

3   Instruction Format Design

Prof. Wawrzynek decides to design a new ISA for his ternary neural network accelerator. He only needs to perform 7 different operations with his ISA: `xor`, `add`, `lw`, `sw`, `lui`, `addi`, and `blt`. He decides that each instruction should be 17 bits wide, as he likes the number 17. There are no `funct7` or `funct3` fields in this new ISA.

3.1   What is the minimum number of bits required for the opcode field?

3.2   Suppose Prof. Wawrzynek decides to make the `opcode` field 6 bits. If we would like to support instructions with 3 register fields, what is the maximum number of registers we could address?

3.3 Given that the `opcode` field is 6 bits wide and each register field is 2 bits
wide in the 17 bit instruction, answer the following questions:

(a) Using the assumptions stated in the above description, how many bits
are left for the immediate field for the instruction `blt` (Assume it takes
`opcode`, `rs1`, rs2, and `imm` as inputs)?

(b) Let n be your answer in part (a). Suppose that `blts` branch imme-
diate is in units of instructions (i.e. an immediate of value 1 means
branching 1 instruction away). What is the maximum number of bits
a `blt` instruction can jump forward from the current `pc` using these
assumptions? Write your answer in terms of n.

(c) Using the assumptions stated in the description, what is the most neg-
ative immediate that could be used in the `addi` instruction (Assume it
takes `opcode`, `rs1`, `rd`, and `imm` as inputs)?

(d) For LUI, we need `opcode`, `rd`, and `imm` as inputs. Using the assumptions
stated in the description, how many bits can we use for the immediate
value?

4 Code Translations

4.1 Assume we have two arrays input and result. They are initialized as follows:

```
int *input = malloc(8*sizeof(int));
int *result = calloc(8, sizeof(int));
for (int i = 0; i < 8; i++) {
    input[i] = i;
}
```

You are given the following RISC-V code. Assume register x10 holds the address of input and register x12 holds the address of result.

```
    add x8, x0, x0
    addi x5, x0, 0
    addi x11, x0, 8
Loop:
    beq x5, x11, Done
    lw x6, 0(x10)
    add x8, x8, x6
    slli x7, x5, 2
    add x7, x7, x12
    sw x8, 0(x7)
    addi x5, x5, 1
    addi x10, x10, 4
    j Loop
Done:
    // exit

    .
```

Please translate this assembly code into C code. Assume that sizeof(**int**) = 4.

4.2   What is the end array stored starting at register x12?