CS315 2025F Lab07 Exam Prob Solutions

Question 1 - C to Assembly

Consider the following C function. Provide and English description of what this function does and provide the RISC-V implementation of this function.

```
int count_rec_c(char *str, char c) {
    int addval = 0;

    if (str[0] == '\0') {
        return 0;
    } else {
        if (str[0] == c) {
            addval = 1;
        }
        return addval + count_rec_c(&str[1], c);
    }
}
```

This function counts the number of occurences of char c in the string str. It computes the count recursively.

```
.global count_rec_s

count_rec_s:
    addi sp, sp, -16
    sd   ra, (sp)
    li   t0, 0              # t0 (addval) = 0
    lb   t1, (a0)
    beq  t1, zero, done     # t1 (str[0]) == '\0' ?
    bne  t1, a1, recstep
    li   t0, 1
recstep:
    sd t0, 8(sp)            # preserve t0
    addi a0, a0, 1          # &str[i]
    call count_rec_s
    ld t0, 8(sp)            # restore t0
    add t0, t0, a0          # t0 (addval) += a0 (retval)
done:
    mv a0, t0               # a0 = t0 (addval)
    ld ra, (sp)
    addi sp, sp, 16
    ret
```

Question 2 - Structs and Assembly ¶

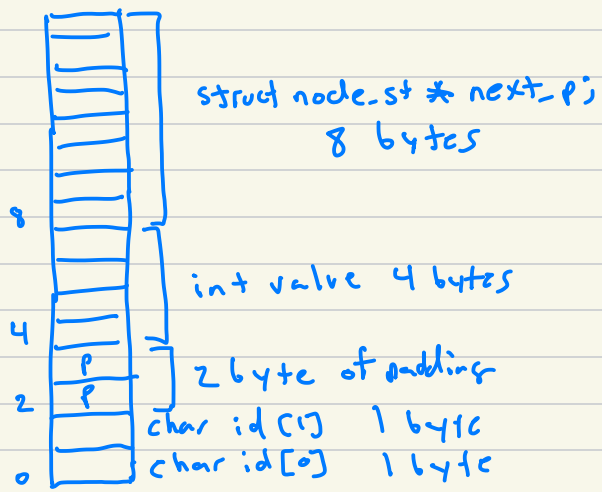
Given what we have learned about how structs are layed out in memory, consider the following struct and answer the following questions.

```
struct node_st {
    char id[2];
    int value;
    struct node_st *next_p;
};
```

How many actual bytes will this struct use in memory when considered alignment and padding?

In RISC-V Assembly, how do you load the `next_p` field value into `t0`?

a)



1 + 1 + 2 + 4 + 8 = 16 bytes total

b) Assume a0 is struct node *np

ld t0, 8(a0) # t0 = np->next_p

Question 3 - RISC-V Machine Code

Lets assume that we have a new RISC-V instruction format called the `x-type`:

```
 31                20 19       15 14    12 11       7 6       0
|   imm[5:0|11:6]     |    rs1    | funct3 |    rd     |  opcode |
```

Assume you have this instruction word in a C `uint32_t` variable called `iw`. Write a C code snippet that can c onstruct the a 32 bit signed immediate value from `iw`, which is a `int32_t` type called `imm`. Your code snipp et should just use C variables and expressions, no function calls. You cannot use `get_bits()` or `sign_extend( )`.

```
uint32_t iw;
uint32_t imm_5_0 = (iw >> 26) & 0b111111;
uint32_t imm_11_6 = (iw >> 20) & 0b111111;
int32_t imm = (imm_11_6 << 6) | imm_5_0;
imm = (imm << 20) >> 20; // sign extend
```

Question 4 - Cache Memory

Assume you have a 2-way Set Associative Case with 32 total words and a block size of 2 words. How many total se ts does this cache have? How many set index bits are there in a 64 bit address?

a) Each set has 2 (ways) × 2 (word per block)
= 4 words
32 (words) / 4 (words) = 8 sets

b) With 8 sets we need 3 set index bits because $2^3 = 8$.