Name: Aarsh Shah

Student Name: Aarsh Shah

Lab Section: B02

Course: Computer Organization – ENCM 369

Lab #: 4

Exercise C: Writing string-handling functions

Copycat

```
#       void copycat(char *dest, const char *src1, const char *src2)
#
        .text
        .globl  copycat
copycat:

        # Students: Replace this comment with appropriate code.
while1:
        lbu     t0, (a1)                # t0 = *src1
        beq     t0, zero, while2        # if (*src1 == zero) goto while2
        sb      t0, (a0)                # *dest = t0
        addi    a0, a0, 1               # dest++
        addi    a1, a1, 1               # src1++
        j       while1
while2:
        lbu     t3, (a2)                # t0 = *src2
        sb      t3, (a0)                # *dest = c
        addi    a0, a0, 1               # dest++
        addi    a2, a2, 1               # src2++
        bne     t3, zero, while2        # if (c != zero) goto while2

        jr      ra

```

lab4reverse

```
lab4reverse:

        # Students: Replace this comment with appropriate code.
        # Prologue
        addi    sp, sp, -32
        sw      ra, 28(sp)
        sw      s3, 24(sp)
        sw      s2, 20(sp)
        sw      s1, 16(sp)
        sw      s0, 12(sp)

        mv      s0, a0

        # Body
        mv      s2, zero                # back = 0

while_start1:
        add     t0, s2, s0              # t0 = &str[back]
        lb      t1, (t0)                # t1 = str[back]
        beq     t1, zero, while_end1    # if (str[back] == zero) goto while_end1
        addi    s2, s2, 1               # back++
        j       while_start1

while_end1:
        addi    s2, s2, -1              # back--
        mv      s1, zero                # front = 0


while_start2:
        ble     s2, s1, while_end2      # if (back <= front) goto while_end2
        add     t0, s2, s0              # t0 = &str[back]
        lb      t1, (t0)                # t1 = str[back]

        add     t2, s1, s0              # t2 = &str[front]
        lb      t3, (t2)                # t3 = str[front]

        addi    s3, t1, 0               # c = str[back]
        sb      t3, (t0)                # str[back] = str[front]
        sb      s3, (t2)                # str[front] = c;

        addi    s2, s2, -1              # back--
        addi    s1, s1, 1               # front++
        j       while_start2

while_end2:
        # Epilogue
        lw      s0, 12(sp)
        lw      s1, 16(sp)
        lw      s2, 20(sp)
        lw      s3, 24(sp)
        lw      ra, 28(sp)
        addi    sp, sp, 32

        jr      ra
```

Exercise E: Programming with logical instructions

write_in_bin:

```
        .text
        .globl  write_in_binary
write_in_binary:

        # Time-saving hint: This is a leaf procedure!
        # Leave str and word in a0 and a1, and
        # use t-registers for local variables.

        li      t0, 0                   # bn = 0
        li      t1, '0'                 # digit0 = '0'
        li      t2, '1'                 # digit1 = '1'
        li      t3, '_'                 # under = '_'

        addi    t4, a0, 39              # &str[39]
        sb      zero, (t4)              # str[39] = '\0'
        li      t5, 38                  # index = 38
        li      t4, 1                   # mask = 1

while_start:

        and     t6, a1, t4              # t6 = word & mask
        bne     t6, zero, else_if       # if (t6 != zero) goto else_if
        add     t6, a0, t5              # &str[index]
        sb      t1, (t6)                # str[index] = digit0
        j       end_if

else_if:
        add     t6, a0, t5              # &str[index]
        sb      t2, (t6)                # str[index] = digit1

end_if:
        addi    t5, t5, -1              # index--
        addi    t0, t0, 1               # bn++
        slli    t4, t4, 1               # mask = mask << 1

        li      t6, 32                  # t6 = 32
        beq     t0, t6, while_end       # if (bn == 32) goto while_end

        li      t6, 3                   # t6 = 3
        and     t6, t0, t6              # t6 = bn & 3
        bne     t6, zero, end_if2
        add     t6, a0, t5              # t6 = &str[index]
        sb      t3, (t6)                # t6 = under
        addi    t5, t5, -1              # index--

end_if2:
        j       while_start

while_end:

        jr      ra
```