iFYP

Lab Manual # 3 RISK V Procedures

NAME: Abdul Hadi Afzal

DATE:15 June 2025

NUST Chip Design Centre (NCDC), Islamabad, Pakistan

Revision History

Revision Number	Revision Date	Revision By	Nature of Revision	Approved By
1.0	04/03/2024	Hira Sohail, Muhammad Bilal	Complete manual	Dr. Waqar
1.1	21/05/2024	Ali Aqdas	Additional Tasks	Dr. Waqar

Contents

Objectives

The objectives of this lab are to:

- Understand basic programs in assembly language covering following o Branches
  - o Loops o Recursion
  - s
- Find the greatest common divisor in assembly language
- Perform sorting using assembly language

Tools

- Venus

A Quick Summary

Example 1 – If-ElseIf-Else Statement

```
_start:

 andi t0, t0, 0          # clear register t0
 andi t1, t1, 0          # clear register t1
 andi t2, t2, 0          # clear register t2
 andi t3, t3, 0          # clear register t3
 andi t4, t4, 0          # clear register t4
 andi t5, t5, 0          # clear register t5
 li t0, 2                # t0 = 2

 li t3, -2       # t3 = -2 slt t1, t0,
zero    # t1 = t0 < 0 ? 1                  : 0
 beq t1, zero, ElseIf # go to ElseIf if t1 = 0
 j EndIf         # end If statement
```

```
ElseIf:
 sgt t4, t3, zero    # t4 = t3 > 0 ? 1 : 0
 beq t4, zero, Else  # go to Else if t4 = 0  j
EndIf         # end Else statement

Else:
 seqz t5, t4, zero    # t5 = t4 == 0 ? 1 : 0

EndIf:
 j EndIf         # end If-ElseIf-Else statement
```

Example 2 – While Loop I

```
_start:
andi t0,   t0, 0           clear register t0
  andi t1, t1, 0           clear register t1
  andi t2, t2, 0           clear register t2
 li t1, 100                t1 = 100

loop:

 add t2, t2, t0   # t2 = t2 + t0
 addi t0, t0, 1          # ++t0
 blt t0, t1, loop        # iterate if t0 < t1


end:
 j end       # end of While loop
```

Example 3 – For Loop I

```
_start:

 andi t0, t0, 0          # clear register t0
 andi t1, t1, 0          # clear register t1

loop:

 andi t2, t2, 0   # clear t2 before starting the loop
 add t1, t1, t0          # t1 = t1 + t0

 addi t0, t0, 1          # ++t0

 slti t2, t0, 100        # t2 = t0 < 100 ? 1 : 0
 bne t2, zero, loop      # go to loop if t2 != 0

end:

 j end               #     end of For loop
```

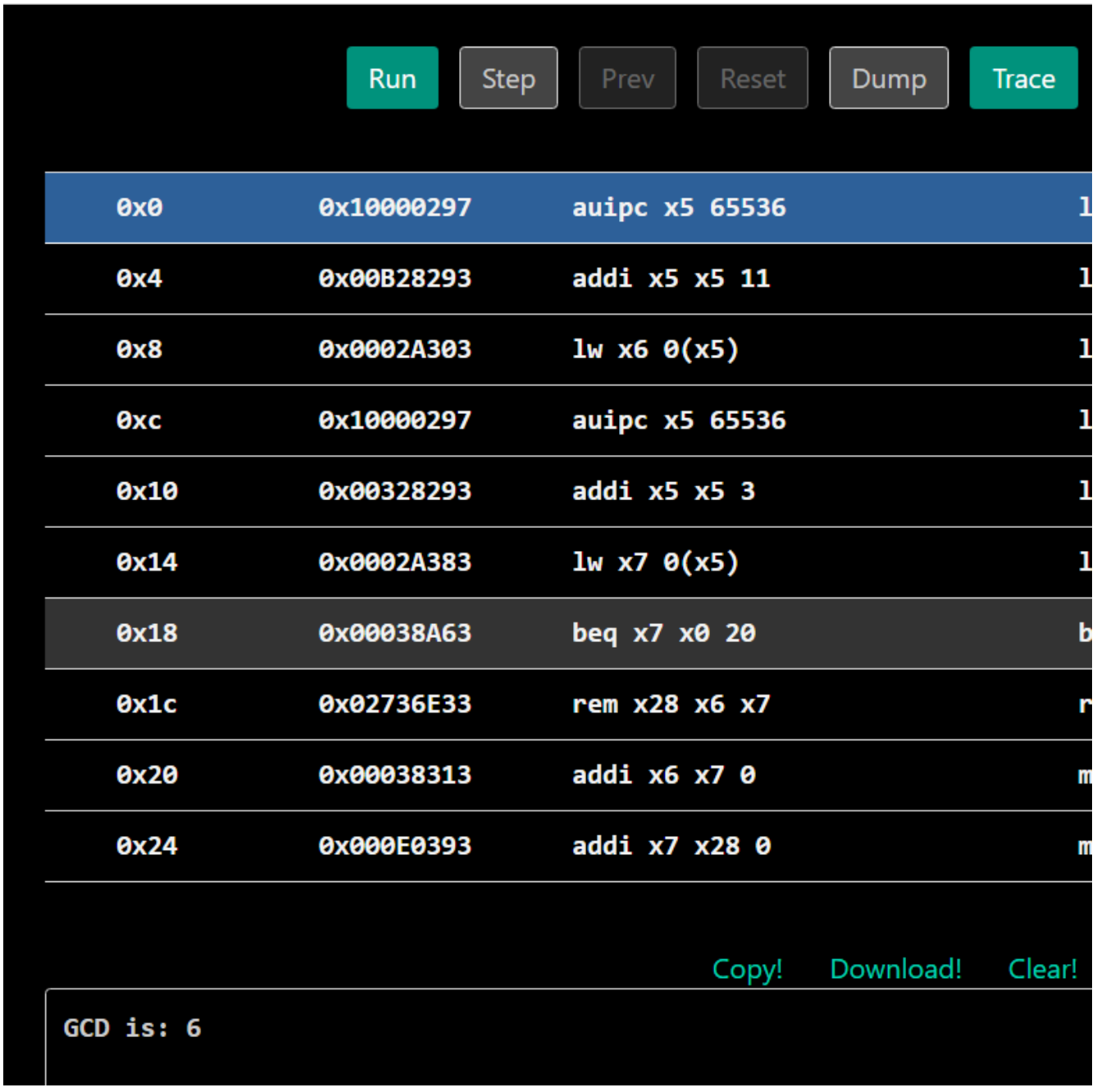
Lab Task 1: Find The Greatest Common Divisor

Code:

```
.data
msg:      .asciiz "GCD is: "
newline:  .asciiz "\n"

a_val:    .word 24      # First number
b_val:    .word 18      # Second number

.text
.globl _start

_start:
# Load a into t1
la t0, a_val
lw t1, 0(t0)

# Load b into t2
la t0, b_val
lw t2, 0(t0)

gcd_loop:
beqz t2, done        # if b == 0, done
rem t3, t1, t2       # t3 = a % b
mv t1, t2            # a = b
mv t2, t3            # b = a % b
j gcd_loop

done:
# t1 now holds the GCD
la a0, msg
jal print_str

mv a0, t1
jal print_digit

la a0, newline
jal print_str

j stop
stop:
j stop

# ===== Utility: Print null-terminated string at address a0 =====
print_str:
mv t4, a0         # t4 = pointer to string
str_loop:
lb t5, 0(t4)      # Load byte
beqz t5, str_ret  # If byte is 0 (null), end
li a0, 11         # syscall: print char
mv a1, t5
ecall
addi t4, t4, 1    # move to next char
j str_loop
```

```
str_ret:
    jr ra

# ===== Utility: Print single digit in a0 =====
print_digit:
    li t6, '0'
    add t5, a0, t6    # Convert number to ASCII
    li a0, 11
    mv a1, t5
    ecall
    jr ra
```

Output:



Lab Task 2: Bubble Sort

CODE:

```
                        .data
array:  .word 12, 5, 8, 1, 19, 7, 3, 15, 2, 10, 6, 4
                    n:      .word 12

                        .text
                      .globl main

                        main:
                         la t0, n
                lw t1, 0(t0)        # t1 = n

        la t2, array        # t2 = base address of array

                li t3, 0            # t3 = i

                      outer_loop:
                    blt t3, t1, outer_continue
                  j print_array       # done sorting

                      outer_continue:
                    li t4, 0            # t4 = j

                  sub t5, t1, t3      # t5 = n - i
                addi t5, t5, -1     # t5 = n - i - 1

                      inner_loop:
                    blt t4, t5, inner_continue
                     addi t3, t3, 1      # i++
                        j outer_loop

                     inner_continue:
                   slli t6, t4, 2      # t6 = j * 4
                 add a0, t2, t6      # a0 = &array[j]
                  lw a1, 0(a0)        # a1 = array[j]

                      addi t6, t6, 4
                add a2, t2, t6      # a2 = &array[j+1]
                 lw a3, 0(a2)        # a3 = array[j+1]

                     ble a1, a3, no_swap

                  # Swap array[j] and array[j+1]
                        sw a3, 0(a0)
                        sw a1, 0(a2)

                        no_swap:
                       addi t4, t4, 1
                        j inner_loop

                     print_array:
                    li t3, 0            # i = 0

                     print_loop:
                    blt t3, t1, print_continue
                          j done

                      print_continue:
```

```
                                slli t4, t3, 2
                                add t5, t2, t4
                                 lw a1, 0(t5)

                            li a0, 1         # syscall print int
                                            ecall

                           li a0, 11        # syscall print char
                                 li a1, 32        # space
                                            ecall

                                 addi t3, t3, 1
                                  j print_loop

                                    done:
                              li a0, 10        # syscall exit
                                            ecall
```

OUTPUT:

Lab Task 3: Quick Sort

CODE:

```
                                    .data
                array:      .word -1, 22, 8, 35, 5, 4, 11, 2, 1, 78
                                array_size: .word 10

                                    .text
                                   .globl main

                                    main:
la s0, array            # s0 = base address of array (never changed)
                      li a0, 0                  # a0 = low index
                   li a1, 9                   # a1 = high index (n - 1)
                    jal quicksort              # sort array from 0 to 9

                     jal print_array          # print sorted array

                        li a0, 10                # syscall: exit
                                         li a1, 0
                                          ecall

                           # -----------------------------
                         # void quicksort(int low, int high)
                                # a0 = low, a1 = high
                           # -----------------------------
                                         quicksort:
              addi sp, sp, -16     # allocate stack space (4 bytes * 4)
                        sw ra, 12(sp)          # save ra at sp+12
                         sw s1, 8(sp)           # save s1 at sp+8
                         sw s2, 4(sp)           # save s2 at sp+4

                                 bge a0, a1, qs_return
```

```
                    mv s1, a0
                    mv s2, a1
                  jal partition

                    mv a0, s1
                    mv a1, a2
                 addi a1, a1, -1
                  jal quicksort

                    mv a0, a2
                 addi a0, a0, 1
                    mv a1, s2
                  jal quicksort

                  qs_return:
        lw ra, 12(sp)        # restore ra
        lw s1, 8(sp)         # restore s1
        lw s2, 4(sp)         # restore s2
   addi sp, sp, 16      # restore stack pointer
                       ret

         # ----------------------------
        # int partition(int low, int high)
           # returns pivot index in a2
         # ----------------------------
                   partition:
                  slli t0, a1, 2
                  add t0, s0, t0
lw t1, 0(t0)             # t1 = pivot = array[high]

    addi t2, a0, -1         # t2 = i = low - 1
      mv t3, a0               # t3 = j = low

                partition_loop:
           bge t3, a1, partition_done

                  slli t4, t3, 2
                  add t5, s0, t4
     lw t6, 0(t5)            # t6 = array[j]

               ble t6, t1, do_swap

                  skip_swap:
                 addi t3, t3, 1
                j partition_loop

                   do_swap:
                 addi t2, t2, 1
                 slli t4, t2, 2
                 add t4, s0, t4

     lw t6, 0(t5)            # t6 = array[j]
     lw t5, 0(t4)            # t5 = array[i]
                sw t6, 0(t4)

                 slli t6, t3, 2
                 add t6, s0, t6
                  sw t5, 0(t6)
```

```
                j skip_swap

              partition_done:
               addi t2, t2, 1
               slli t4, t2, 2
               add t4, s0, t4

               slli t5, a1, 2
               add t5, s0, t5
                lw t6, 0(t5)
                lw t5, 0(t4)
                sw t6, 0(t4)

               slli t6, a1, 2
               add t6, s0, t6
                sw t5, 0(t6)

mv a2, t2              # return pivot index
                ret

       # ----------------------------
            # void print_array()
       # ----------------------------
                print_array:
     li t0, 0               # index = 0

                print_loop:
    li t2, 10              # array size
           bge t0, t2, done_print

               slli t3, t0, 2
               add t4, s0, t3
   lw a1, 0(t4)            # a1 = array[i]
 li a0, 1               # syscall: print_int
                ecall

li a0, 11              # syscall: print_char
       li a1, 32              # space
                ecall

               addi t0, t0, 1
                j print_loop

                done_print:
li a0, 11              # syscall: print_char
      li a1, 10              # newline
                ecall
                 ret
```

OUTPUT:

0x0	0x10000417	auipc x8 65536	la s0, array # s0 = base address of array (never changed)
0x4	0x00040413	addi x8 x8 0	la s0, array # s0 = base address of array (never changed)
0x8	0x00000513	addi x10 x0 0	li a0, 0 # a0 = low index
0xc	0x00900593	addi x11 x0 9	li a1, 9 # a1 = high index (n - 1)
0x10	0x014000EF	jal x1 20	jal quicksort # sort array from 0 to 9
0x14	0x0F0000EF	jal x1 240	jal print_array # print sorted array
0x18	0x00A00513	addi x10 x0 10	li a0, 10 # syscall: exit
0x1c	0x00000593	addi x11 x0 0	li a1, 0
0x20	0x00000073	ecall	ecall

Copy! Download! Clear!

```
-1 1 2 4 5 8 11 22 35 78
```