Contents

Instructions for Implementation

Each group of two students completes two randomly assigned projects, with each student taking responsibility for one project while understanding the other. Group members should agree on their respective tasks.

Evaluation Method

- Oral Presentation: Simulate on RARS, present source code, and answer questions.
- Report Submission: Each group submits a printed report including analysis of methods, algorithms, source code, and simulation results.
- Program Submission: Submit the report and assembly program in soft copy. Follow this file naming convention:

nxx_gyy_StudentName.asm

+ xx: Project number. E.g., Project 8 -> xx = 08.
+ yy: Group number. E.g., Groudp 9 -> yy = 09.
+ StudentName: Full name in Vietnamese without accents or spaces, with capitalized initials. E.g., Nguyễn Văn A -> NguyenVanA.
+ Example file: Lê Mạnh Hùng, Group 21, Project 8 → n08_g21_LeManhHung.asm.

Notes for Implementation:

- The program should have a menu interface (if needed) and be repeatable without restarting.
- Data inputs should be validated.
- Source code should be modularized into subprograms (if needed).
- Write clear, well-commented, and neatly formatted source code.

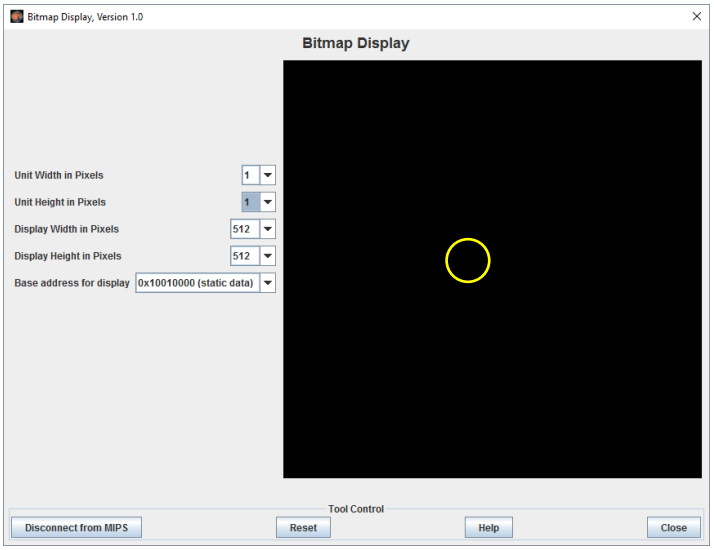
1. Drawing Images on a Bitmap Screen

Write a program to draw a circular ball moving on a bitmap screen (Bitmap Display). The ball should reverse direction when hitting screen edges.

Requirements:

- Set screen dimensions to 512x512 with a 1x1 pixel size.
- Movement depends on user input (W: up, S: down, A: left, D: right, Z: increase speed, X: decrease speed). Use Keyboard and Display MMIO Simulator.
- Initial position of the ball is at the center of the screen.

Hint: *Move the object by erasing it at its old position (using the background color) and redrawing it at the new position.*

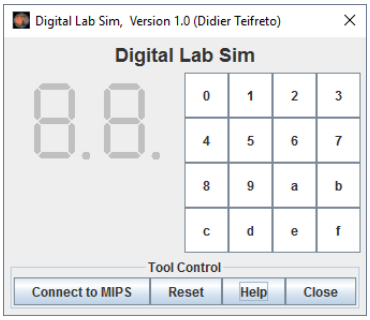


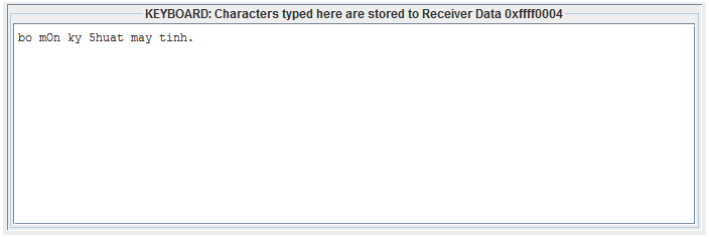
2. Testing Typing Speed and Accuracy

Write a program to measure typing speed and display results on two 7-segment displays.

Principles:

- Provide a text sample in the source code. Example: "bo mon ky thuat may tinh".
- Use the timer (Digital Lab Sim) to measure the time between interrupts.
- The user inputs characters from the keyboard (Keyboard and Display MMIO Simulator). For example, typing "bo mOn ky 5huat may tinh", the program must count correct characters (in this example, the user mistyped ‘O’ and ‘5’). Display the count on the 7-segment displays.
- Calculate typing speed: the completion time and the number of words per time unit.





3. Infix to Postfix Expression

Write a program to compute the value of any expression using the postfix traversal.

Requirements:

1. Input an infix expression, e.g., 9 + 2 + 8 * 6
2. Output the postfix expression, e.g., 9 2 + 8 6 * +
3. Calculate and display the value of the entered expression.

Constraints:

- Constants are integers between 0 and 99.
- Operators include addition (+), subtraction (-), multiplication (x), division (/), modulo (%), and parentheses.

4. Memory Allocation Function *malloc()*

The provided program includes the *malloc()* function and examples in RISC-V assembly to allocate memory dynamically for a pointer variable.

Tasks:

1) Fix the error in word/array memory allocation (ensure word addresses are multiples of 4).
2) Write a function to retrieve the value of a pointer.
3) Write a function to get the address of a pointer.
4) Write a function to copy characters from one string to another string using pointers.
5) Write a function to release memory allocated for pointers.
6) Write a function to calculate the total allocated memory.
7) Write *malloc2* function to allocate 2D arrays with *.word* type, including parameters for:
    a. Starting address of the array
    b. Number of rows
    c. Number of columns
8) In addition to 7), write *getArray[i][j]* and *setArray[i][j]* functions to get/set the value of an array element at row *i* and column *j*.

Sample program:

```
.data
CharPtr: .word  0 # Bien con tro, tro toi kieu asciiz
BytePtr: .word  0 # Bien con tro, tro toi kieu Byte
WordPtr: .word  0 # Bien con tro, tro toi kieu Word
```

```
.kdata
# Bien chua dia chi dau tien cua vung nho con trong
Sys_TheTopOfFree: .word  1
# Vung khong gian tu do, dung de cap bo nho cho cac bien con tro
Sys_MyFreeSpace:

.text
     #Khoi tao vung nho cap phat dong
     jal   SysInitMem

     #-----------------------
     #  Cap phat cho bien con tro, gom 3 phan tu,moi phan tu 1 byte
     #-----------------------
     la   a0, CharPtr
     addi a1, zero, 3
     addi a2, zero, 1
     jal malloc
     #-----------------------
     #  Cap phat cho bien con tro, gom 6 phan tu, moi phan tu 1 byte
     #-----------------------
     la   a0, BytePtr
     addi a1, zero, 6
     addi a2, zero, 1
     jal  malloc
     #-----------------------
     #  Cap phat cho bien con tro, gom 5 phan tu, moi phan tu 4 byte
     #-----------------------
     la   a0, WordPtr
     addi a1, zero, 5
     addi a2, zero, 4
     jal  malloc


lock: j lock
      nop

#-----------------------------------------
#  Ham khoi tao cho viec cap phat dong
#  @param    khong co
#  @detail   Danh dau vi tri bat dau cua vung nho co the cap phat duoc
#-----------------------------------------
SysInitMem:  la   t9, Sys_TheTopOfFree  #Lay con tro chua  dau tien
con trong, khoi tao
             la   t7, Sys_MyFreeSpace   #Lay dia chi dau tien con
trong, khoi tao
              sw   t7, 0(t9)             # Luu lai
```

```
            jr   ra

#-----------------------------------------
#  Ham cap phat bo nho dong cho cac bien con tro
#  @param  [in/out]   $a0   Chua dia chi cua bien con tro can cap phat
#                           Khi ham ket thuc, dia chi vung nho duoc
cap phat se luu tru vao bien con tro
#  @param  [in]       $a1   So phan tu can cap phat
#  @param  [in]       $a2   Kich thuoc 1 phan tu, tinh theo byte
#  @return            $v0   Dia chi vung nho duoc cap phat
#-----------------------------------------
malloc:   la   t9, Sys_TheTopOfFree   #
          lw   t8, 0(t9)     #Lay dia chi dau tien con trong
          sw   t8, 0(a0)     #Cat dia chi do vao bien con tro
          addi v0, t8, 0     #Dong thoi la ket qua tra ve cua ham
          mul  t7, a1, a2    #Tinh kich thuoc cua mang can cap phat
          add  t6, t8, t7   #Tinh dia chi dau tien con trong
          sw   t6, 0(t9)     #Luu tro lai dia chi dau tien do vao bien
Sys_TheTopOfFree
          jr   ra
```

5. Syntax Checker for Assembly Language Instructions

The processor's compiler checks the syntax of assembly instructions before translating them into machine code. Write a program to verify the syntax of assembly instructions (excluding pseudoinstructions).

Requirements:

- Input an assembly instruction from the keyboard (Keyboard and Display MMIO Simulator). Example: *beq s1, 31, t4*
- Verify whether the opcode of the instruction is correct. In the example, the opcode *beq* is valid, so display: *"opcode: beq, hợp lệ"*
- Check the validity of the operands. In the example, *s1* is valid, *31* is invalid, and further checks on *t4* are unnecessary once an error is found.

Hint: *Build a structure containing the format of each instruction, including the instruction name and operand types for operands 1, 2, and 3.*

6. RAID 5 Disk Simulation

RAID 5 disk systems require at least three hard drives, with parity data distributed across the disks. Write a program to simulate the operation of RAID 5 with three disks. Assume each data block contains four characters.

Requirements:

- Input a string of characters (length must be a multiple of 8).
  (Example: DCE.****ABCD1234HUSTHUST)
- Divide the input string into 4-byte blocks.
  - Block 1 (DCE.) is stored on Disk 1.

- Block 2 (****) is stored on Disk 2.
- Parity data for these blocks is stored on Disk 3, calculated using XOR of ASCII values.

```
Nhap chuoi ki tu : DCE.****ABCD1234HUSTHUST
     Disk 1                Disk 2                Disk 3
 --------------        --------------        --------------
|     DCE.     |      |     ****     |      [[ 6e,69,6f,04]]
|     ABCD     |      [[ 70,70,70,70]]      |     1234     |
[[ 00,00,00,00]]      |     HUST     |      |     HUST     |
 --------------        --------------        --------------
```

7. Drawing Images with ASCII Characters

Display an ASCII art image using console or display interface.

Example:

An image of DCE with borders and colored numbers is provided.

```
                                          *************
 **************                          *3333333333333*
 *222222222222222*                        *33333********
 *22222******222222*                      *33333*
 *22222*      *22222*                     *33333********
 *22222*       *22222*      *************  *3333333333333*
 *22222*       *22222*    **11111*****111* *33333********
 *22222*       *22222*  **1111**       **  *33333*
 *22222*      *222222*  *1111*             *33333********
 *22222*******222222*  *11111*             *3333333333333*
 *2222222222222222*    *11111*              *************
 ***************       *11111*
       ---              *1111**
     / o o \             *1111****   *****
     \   > /              **111111***111*
      -----                 ***********    dce.hust.edu.vn
```

Tasks:

- Display the image on the console or display interface (Keyboard and Display MMIO Simulator).
- Modify the image to leave only the outline of DCE (no colored numbers inside).
- Rearrange the letters to display ECD instead of DCE. Attached decorations may also be rearranged.
- Input colored numbers for the letters D, C, E from the keyboard and display the image with new colored numbers.

Note: *Do not allocate new memory for adjusted images; use the existing memory space.*

8. Pocket Calculator

Use a keypad and two 7-segment displays (Digital Lab Sim) to build a simple calculator supporting operations +, -, *, /, and % on integers.

Requirements:

- Use keypad keys for operations:

- **a** for addition (+)
- **b** for subtraction (–)
- **c** for multiplication (*)
- **d** for division (/)
- **e** for modulo (%)
- **f** for equals (=)

- When entering a number, display the last two digits of the number on the two 7-segment displays. Example of entering 123: 1 → 01, 2 → 12, 3 → 23.
- After entering a number, select an operation (+, –, etc.) and press **f** to calculate.
- Allow consecutive calculations (like the Windows Calculator app).

9. Testing Sorting Algorithms

Tasks:

- Learn how to read data from files.
- Create an interface for users to input the filename containing numbers to sort (with large datasets pre-prepared).
- Allow users to choose a sorting algorithm: Bubble Sort, Insertion Sort, or Selection Sort. (Extra credit for implementing additional algorithms).
- Run the selected algorithm and display the execution time.
- Write the sorted results back to a file.

10. Electronic Lock

Use a keypad and two 7-segment displays (Digital Lab Sim) to simulate an electronic lock system with the following requirements:

- The password consists of n digits (at least 4 digits, ranging from 0 to 9).
- To unlock the door:
  - Enter the password using the matrix keypad and end with the '**f**' key.
  - If the password is correct, the lock opens, and the 7-segment displays ON.
  - If the password is incorrect, the 7-segment displays OF(F).
- If the wrong password is entered three consecutive times, the lock will freeze for 1 minute, during which no key presses will work.
- To change the password:
  - Press the '**a**' key, then input the current password and press '**f**'.
  - If the current password is correct, the system will prompt for a new password.

11. Plotting Function Graphs

Plot mathematical functions on a bitmap screen (Bitmap Display).

Requirements:

- Input coefficients for the equation $\mathbf{y = ax^2 + bx + c}$ from the keyboard (i.e., **a**, **b**, **c**).
- Input the graph color.
- Plot the graph with the coordinate axes (Oxy) centered on the 512x512 bitmap screen.
- After plotting one graph, allow the user to choose to plot additional graphs (overlaid on the existing ones) or end the program.

12. Reading BMP Files and Displaying on Bitmap Screen

Tasks:

- Learn about system functions for opening and reading files.
- Read BMP image files with a maximum resolution of 512x512.
- Display the image on a bitmap screen (Bitmap Display).

13. Memory Training Game

Refer to: https://www.memozor.com/simon-games/simon-game

Tasks:

- Learn about system functions for generating random numbers.
- Display four squares of different colors (IDs 1 to 4) on the Bitmap Display.
- The first round starts with one square randomly highlighted (brighter color) for 1 second before returning to its normal color.
- The player selects the square by pressing the corresponding key on the keypad, keys 1 to 4, (Digital Lab Sim).
- If the player inputs the key correctly, the second round starts with two squares randomly highlighted.
- If the player inputs the two keys correctly in sequence, the thrid round starts with three squares randomly highlighted, and so on.
- If the player inputs the keys incorrectly, the game ends.

14. Card-Flipping Game

Refer to: https://www.memozor.com/memory-games/for-toddlers-babies/dick-bruna

Tasks:

- Learn about system functions for generating random numbers.
- Display a 4x4 grid of square cards with 8 pairs of colors arranged randomly in the face-down state in a bitmap screen (Bitmap Display).
- The user flips two cards by pressing two keys on the keypad (Digital Lab Sim). The corresponding cards are revealed.
- If the two flipped cards have the same color, they remain the face-up state. If the colors differ, they flip back down.
- The game ends when all cards are flipped face-up.

15. Number Memory Game

Refer to: https://www.memozor.com/other-memory-games/numbers-memory-games/grid-of-numbers-to-remember-1-1000

Tasks:

- Learn about system functions for generating random numbers.
- Display random numbers on the display (Keyboard and Display MMIO Simulator).
- After a countdown timer, erase the numbers.
- The user inputs the numbers via the keyboard, separating them with spaces and ending with Enter.
- The program checks whether the entered numbers are correct (not need to be in order). If correct, the next round begins. If incorrect, the game ends.

16. Script-Based Music Playback – Electronic Keyboard

Tasks:

- Learn about system functions for playing music.
- A music track is represented as a string of parameters: pitch, duration, instrument, and volume. Example: "60, 1200,1, 120, 73, 220,1, 125, ...".
- Prepare four preset music tracks.
- During execution, the user selects a music track by pressing keys 1 to 4. Pressing 0 stops playback.

17. Digital Clock

Display the current time using two 7-segment displays (Digital Lab Sim).

Requirements:

- Press keys on the keypad (Digital Lab Sim) to toggle display modes:
  - 1: Show hours
  - 2: Show minutes
  - 3: Show seconds
  - 4: Show the day
  - 5: Show the month
  - 6: Show the last two digits of the year
- Update and display the time every second.
- Play a sound every full minute.

Hint: *Learn about system functions for retrieving the current time and playing sounds.*