PPC5

ID: **111061894**
Name: **Snehit**

Setup

Software	Purpose
Cygwin (3.5.4)	For Unix-like Environment
SDCC (4.4.0)	compiler suite that targets the Intel MCS51 based microprocessors
Notepad++ (8.7.1)	Write and edit .c files
EdSim51DI (2.1.36)	Simulator for 8051

Table 1: Setup describing Software with respective version and purpose.

Creating and Compiling Makefile

Using same Makefile from CP4 with modifying filenames as “test3threads” to “testlcd” and adding “lcdlib.rel buttonlib.rel keylib.rel” as OBJ so that it can be compatible with CP5 Part 1. For Part 2, use the Markfile of Part 1 replacing “testlcd” by “dino”. Running the following commands in Cygwin (3.5.4)

```
$ make clean
$ make
```

as shown in Fig. 1. *make clean* will clear the files generated from previous execution (if any) and then *make* command will create new require file as per the code written in *.c* files. Table 2 shows the result of respective make command. Note, screenshots (Fig. 1 and Table 2) below are for test3threads using Approach 2 (discussed onward).

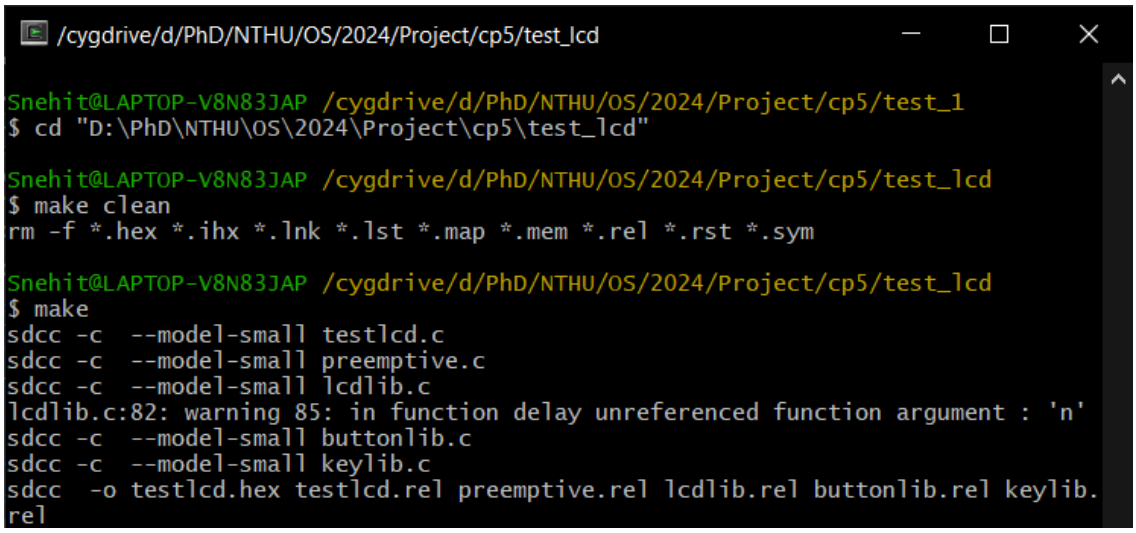


Fig. 1: Screenshot of Cygwin after running *make clean* and *make* command.

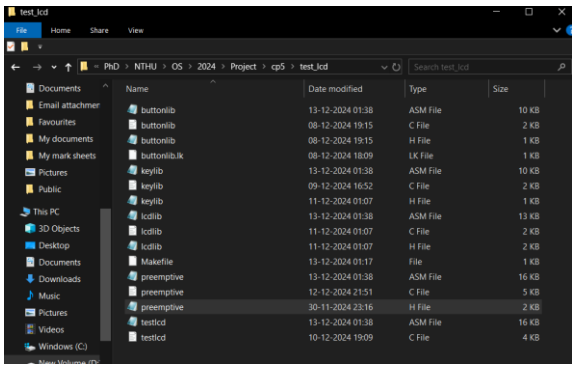
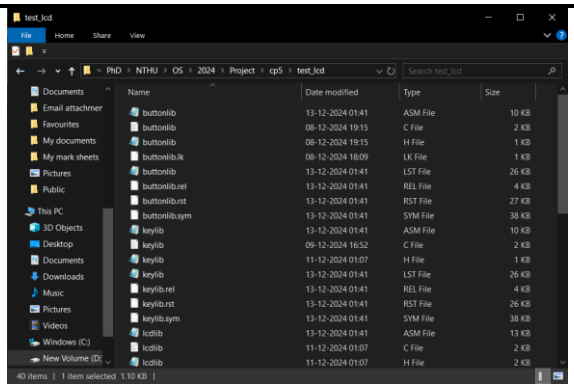
After *$ make clean*	After *$ make*
	

Table 2: results of Makefile compilation

Button bank:

```
void Producer1(void)
{
    currentButton = '/0';
    while (1)
    {
        while(!AnyButtonPressed());
        currentButton = ButtonToChar();
        while(AnyButtonPressed());

        SemaphoreWait(empty);

        SemaphoreWait(mutex);
        sharedBuffer[head] = currentButton;
        head = (head==BUFFER_SIZE-1) ? 0 : head+1;
        SemaphoreSignal(mutex);

        SemaphoreSignal(full);
    }
}
```

Fig 2: Code snippet of Producer1 to access Buttons

```
void Consumer(void)
{
    // Configure serial port for polling mode
    TMOD |= 0x20;    // Timer1 mode 2: 8-bit auto-reload
    TH1 = 0xFA;      // (Hex) Baud rate 4800 for 11.0592 MHz or TH1=-
    SCON = 0x50;     // Mode 1: 8-bit UART, REN enabled
    TR1 = 1;         // Start Timer1

    while (1)
    {
        SemaphoreWait(full);

        SemaphoreWait(mutex);
        sb = sharedBuffer[tail];
        tail = (tail==BUFFER_SIZE-1) ? 0 : tail+1;
        SemaphoreSignal(mutex);

        SemaphoreSignal(empty);

        while(!LCD_ready());
        LCD_write_char(sb);

    }
}
```

Fig 3: Code snippet of Consumer to display the characters on LCD panel.

The character from the pressed button is fetched from ButtonToChar() which is then transferred to sharedBuffer at memory location 0x2C to0x2E (i.e. we have Buffer of size 3) which is shown in Fig. 2 that is Producer1 code snippet. This character on sharedBuffer then displayed on LCD using LCD_write_char() as shown in Fig. 3 Consumer code snippet. The result of testlcd code in execution for testing all buttons from 0 to 7 is as shown in Table 3.

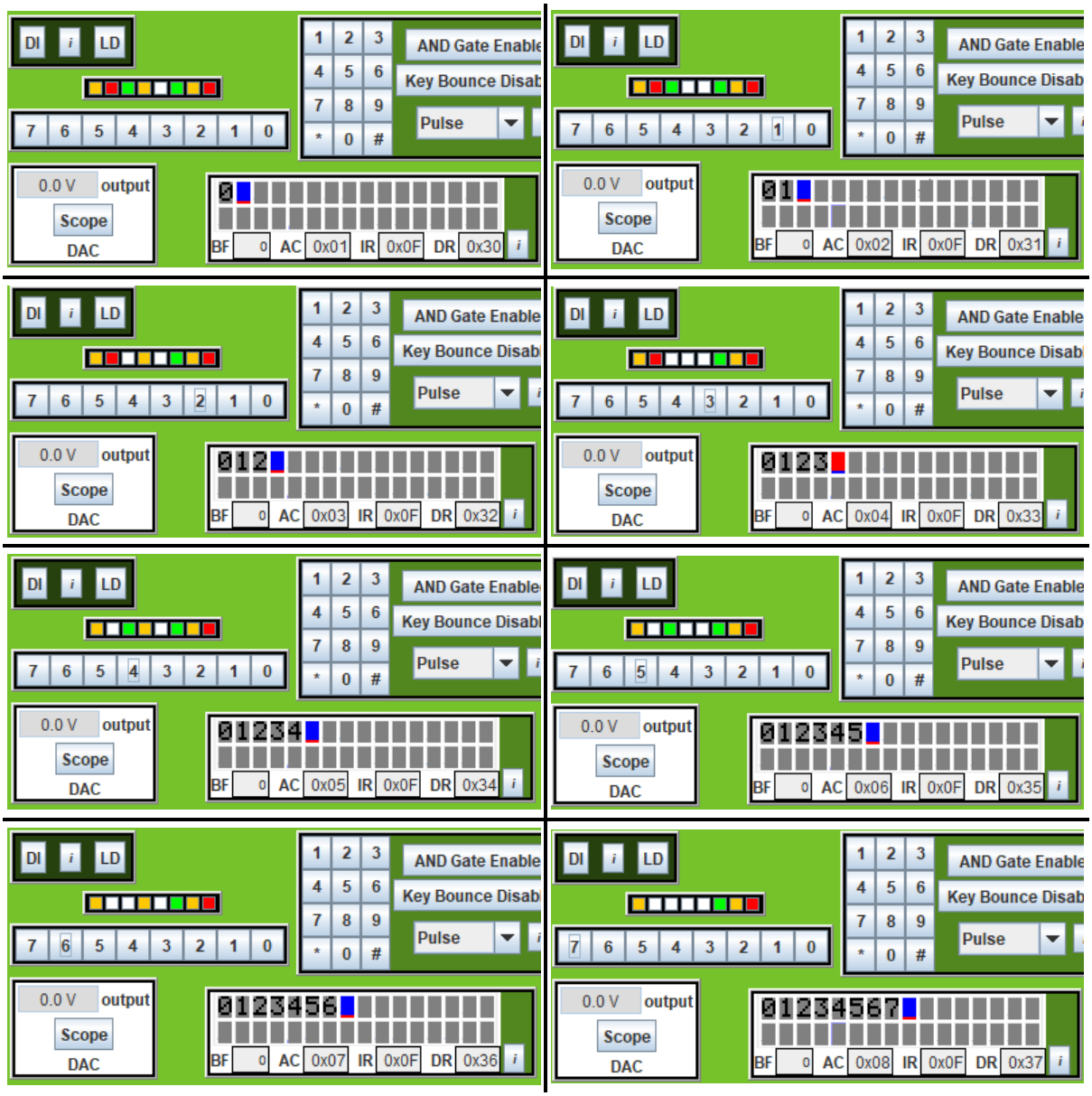


Table 3: Button bank input from button “0” to “7” with results displayed on LCD

Keypad:

The character from the pressed key is fetched from KeyToChar() which is then transferred to sharedBuffer refer Fig. 4. Then similarly to Button section, the character on sharedBuffer then displayed on LCD using LCD_write_char() as shown in Fig. 3 Consumer code snippet. The result of testlcd code in execution for testing all keys from 0 to 9 with * and # is as shown in Table 4.

```
void Producer2(void)
{
    currentKey = '/0';
    while (1)
    {
        while(!AnyKeyPressed());
        currentKey = KeyToChar();
        while(AnyKeyPressed());

        SemaphoreWait(empty);

        SemaphoreWait(mutex);
        sharedBuffer[head] = currentKey;
        head = (head==BUFFER_SIZE-1) ? 0 : head+1;
        SemaphoreSignal(mutex);

        SemaphoreSignal(full);
    }

}
```

Fig 4: Code snippet of Producer2 to access Keys

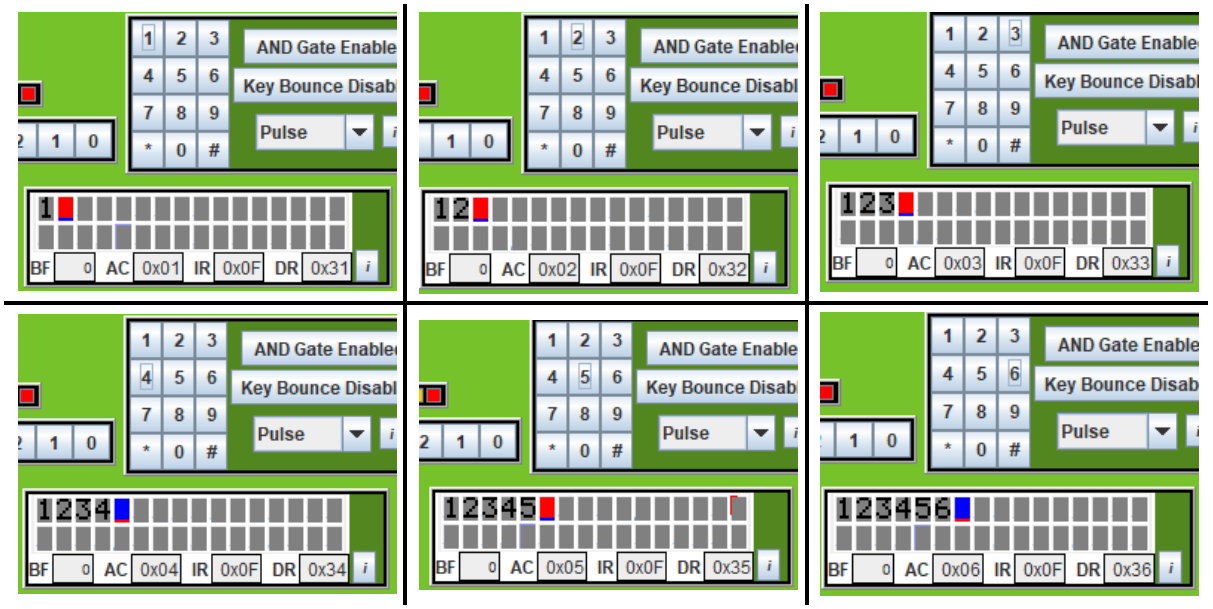


Table 4: Keypad input from top left key "1" to bottom right key "#" with results displayed on LCD

Dino:

The Producer2 from testlcd is used for "keypad_ctrl" shown in Fig. 5, any key pressed on keypad will be transferred to sharedBuffer which will be then used in "render_task" and "game_ctrl".

```
void keypad_ctrl(void)
{
    currentKey = '/0';
    while (1)
    {
        while(!AnyKeyPressed());
        currentKey = KeyToChar();
        while(AnyKeyPressed());

        SemaphoreWait(empty);

        SemaphoreWait(mutex);
        sharedBuffer[head] = currentKey;
        head = (head==BUFFER_SIZE-1) ? 0 : head+1;
        SemaphoreSignal(mutex);

        SemaphoreSignal(full);
    }
}
```

Fig 5: Code snippet of keypad_ctrl to access Keys

```
if (status==2) {
    for (j=0; j<8; j++) {

        temp = 0x01<<(7-j);

        LCD_cursorGoTo(0, j);
        if (dino_loc==0 & j==1) LCD_write_char('#');
        else if (!(Pannel_0_0 & temp)) LCD_write_char('*');
        else LCD_write_char(' ');

        LCD_cursorGoTo(0, j+8);
        if (!(Pannel_0_8 & temp)) LCD_write_char('*');
        else LCD_write_char(' ');

        LCD_cursorGoTo(1, j);
        if (dino_loc==1 & j==1) LCD_write_char('#');
        else if (!(Pannel_1_0 & temp)) LCD_write_char('*');
        else LCD_write_char(' ');

        LCD_cursorGoTo(1, j+8);
        if (!(Pannel_1_8 & temp)) LCD_write_char('*');
        else LCD_write_char(' ');
    }
    Pannel_0_0 = (Pannel_0_0<<1) | (Pannel_0_8 >> 7);
    Pannel_0_8 = (Pannel_0_8<<1) | (Pannel_1_0>>7);

    Pannel_1_0 = (Pannel_1_0<<1) | (Pannel_1_8 >> 7);
    Pannel_1_8 = (Pannel_1_8<<1) | (Pannel_0_0>>7);
    score += 1;

    if (!dino_loc & Pannel_0_0>>7) status=3;
    if (dino_loc & Pannel_1_0>>7) status=3;
}
```

Fig 6: Code snippet of game logic from render_task (#' is used for dinosaur and '*' for cactus)

The game logic for generating cactus map depending on the cactus and dinosaur location set the status of game. Here status 1 is for initialized state of game, status 2 is the game in execution and 3 is when game is over. Note that during multiple trials, the characters of cactus and dinosaur were unable to displayed on LCD. To compensate that I have choose "#" for dinosaur and "*" for cactus.

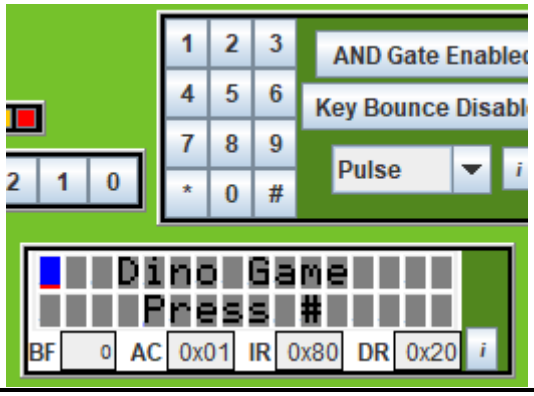
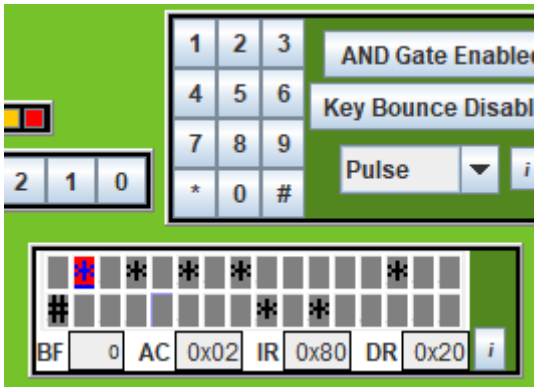
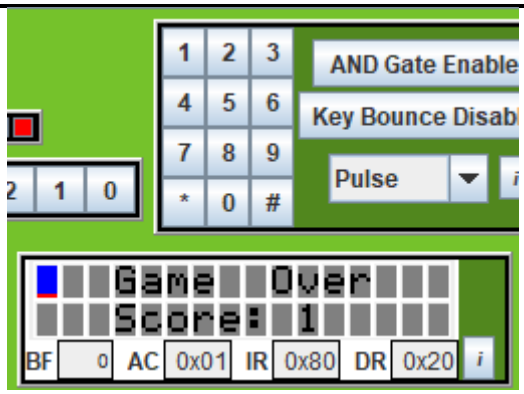
Status 1: LCD is initialized, the instruction is given to press # to begin the game.	
Status 2: Screenshot of game where '*' are for cactus and '#' for dinosaur'. Please note that the game is still not working properly and there are many sections to improve in code, but within the given submission deadline this is what I have managed. The Cactus '*' will be shifting towards left every cycle one by one. And once it hit the dinosaur '#' it'll change the state of game.	
Status 3: The final state where the Score will be displayed with the message of 'Game Over'	

Table 5: Game in execution with screenshot at each state