EECS 322: SPIM assignment / project

All code must be commented!

Each problem part (1,2,3a,3b,…) will be in a separate file: problem_1.s ….

You may be asked to demonstrate your program.

You must turn in floppy and printouts.

1. **(10%) Problem A.7 on page A-77 in textbook**
2. **(20%) Problem A.8 on page A-77 in textbook**

3a. (10%) Modify the mapped_io.s program to echo (tx) each rx character back as is typed. Read A-36 to A-38

EECS 322: SPIM assignment / project

Note: functions return values via $v0
if function uses $s0 to $s7 it must be saved on the stack. (see page 134 and page A-22)

3b. (10%) Improve the mapped_io.s by writing your own ANSI C Language function: char *gets(char *s)

where
char *s is a pointer to a pre-allocated string of bytes.

Gets returns the original pointer *s passed in.

Gets inputs each character and echos it until a newline is encountered (0x0a). The newline is not saved in the final string. The returned string is null terminated.

EECS 322: SPIM assignment / project

3c. (10%) Improve the mapped_io.s by writing your own ANSI C Language function: int puts(char *s)

where

char *s is a pointer to a string of bytes to be printed.

Puts prints each character until a null is encountered (0x0a) in the string. A newline is then also printed to the console.

Puts returns the number of characters written to the console.

EECS 322: SPIM assignment / project

3d. (10%) Write your own ANSI C Language function:

int atoi(char *s)

where

char *s is a pointer to a null terminated string of bytes of decimal ascii digits.

atoi returns the integer (I.e. convert to binary) of the input string.

3e. (10%) Write tour own C Language function:

void itoa(char *s, int n);

where

int n is a binary integer

char *s is a pointer to a null terminated string of bytes of decimal ascii digits converted from n.

EECS 322: SPIM assignment / project

3f. (20%) Rewrite problem 1 using your own subroutines.

No system calls allowed.

Also hand in the C language version of your program. You do not need to run the C code.

char *gets(char *s) reads until newline, newline discarded, and returns a string terminated with a zero.

int puts(char * s) prints a string followed by newline and returns the number of characters written.

int atoi(char *s) converts a ascii string to binary number

void itoa(char *s, int n) returns a string converted from n

EECS 322: SPIM assignment / project

```
#Goto to dos prompt and type: pcspim -mapped_io
#Warning bugs have been inserted!
          .globl main
main:                                   #main has to be a global label
          addu      $s7, $0, $ra        #save the return address in a global register

                              #Output the string "Hello World" on separate line
          .data
#         .globl    hello
hello:    .asciiz "\nHello World\n"     #string to print
goodbye:  .asciiz "\nGoodbye\n"
rx_buffer: .asciiz "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx\n"
rx_cntl:  .word     0xffff0000
tx_cntl:  .word     0xffff0008

          .text
          li        $v0, 4                                  #print_str (system call 4)
          la        $a0, hello                              # takes the address of string as an
argument
          syscall

          la        $a0, rx_buffer
rx_wait:
          lw        $t1,rx_cntl
rx_wait1:
          lw        $t2,0($t1)                  # ready?
          andi      $t2,$t2,1
          beq       $t2,$0,rx_wait1             #no - loop
          lw        $t2,4($t1)                  #yes - get character
          sb        $t2,0($a0)                  #..store it
          addi      $t2,$t2,-10                           #end of line?
          beq       $t2,$0,rx_wait2             #yes - make it zero
          addi      $a0,$a0,2                   #increment string address
          j         rx_wait1
```

EECS 322: SPIM assignment / project

```
rx_wait2:
          sb        $0,0($a0)                       #store zero

          la        $a0, rx_buffer
tx_wait:
          lw        $t1,tx_cntl
tx_wait1:
          lw        $t2,0($t1)
          andi      $t2,$t2,1
          beq       $t2,$0,tx_wait1
          lbu       $t2,0($a0)
          beq       $t2,$0,tx_wait2
          sw        $t2,4($t1)
          addi      $a0,$a0,1                       #increment string address
          j         tx_wait1
tx_wait2:

                                        #Usual stuff at the end of the main
          addu      $ra, $0, $s7                    #restore the return address
          jr        $ra                                     #return to the main program
          add       $0, $0, $0                              #nop
```