CS 150

Lab 05 - Logic Operations

Prof. Manwaring

Colby Blair

cblair@vandals.uidaho.edu

TA: Xiao He (not sure on the first name)

Due 03/26/09 11:59 PM

I. Decomposition

This is the basic logic of the program, in pseudo code:

```
Set R6 to the HEAD of the stack
Prompt for A string input
INPUT routine, passed argument mem location for ASTRING
(INPUT routine)
    Push all registers to the stack
    Save R6 to SAVEDR6 label
    R6 = 16d
    Read char from input
    If input = newline (x0A), branch ahead to restoring R6
    If ASCII val >< 0 or 1, branch down to beep, and then branch back to input
    Else, echo input, and store input in mem location of R1
    Increment R1
    Decrement R6
    If decremented R6 = 0, exit branch. Otherwise, branch back to reading
        char from input
    Restore R6 to point to beginning of stack
    Restore the registers, pop from the stack
    RETurn
Prompt for B string input
INPUT routine, passed argument mem location for BSTRING
(INPUT routine)
SIGNEXT (Sign extend) both A and B
    (SIGNEXT routine)
    Push all registers to the stack
    Count how many chars are in the string, and store count in R5
    Set R2 to R1
    Put the string pointed to by R1 in CSTRING mem location
    Set all the chars in string pointed to by R1 to the MSB in CSTRING
        R2 is now at the end of the string in R1; move it back count-many times
    Set the chars in CSTRING to the current location of R2
    Restore the registers, pop from the stack
NAND (output NAND operation) for A and B
    (NAND routine) - R1 set to first string, R2 to second
    Push all registers to stack
    If R1 AND R2, ouput '0'
```

    Else output '1'
    Pop all registers from stack
NOR (output NOR operation) for A and B
(NOR routine) - R1 set to first string, R2 to second
    Push all registers to stack
    If R1 OR R2, ouput '0'
    Else output '1'
    Pop all registers from stack
XOR (output XOR operation) for A and B
(XOR routine) - R1 set to first string, R2 to second
    Push all registers to stack
    If R1 XOR R2, ouput '0'
    Else output '1'
    Pop all registers from stack

II Data

See attached flow chart.

Code (see attached as well for better formatting)

III. Output

Enter A: 10011
Enter B: 11110000
A NAND B: 0000000000001111
A NOR B: 0000000000001100
A XOR B: 0000000000000011

Enter A: 0011110010101010
Enter B: 0110101011
A NAND B: 1111111101010101
A NOR B: 1100001001010100
A XOR B: 0011110100000001

IV. Conclusion

This was a massive assignment. Not to say at all that is wasn't a good assignment. The amount of code turned out to be way more massive than I ever anticipated. I was more successful in using subroutines here than the other programs. Mostly because using labels and thus subroutines was pretty much manditory on this assignment.

As I began coding, I kept the code pretty clean and portable. If I needed, I could use these subroutines for other programs pretty easy. However, as the deadline wound down, my programming got a little uglier and less clear, which makes time troubleshooting in the last hours harder. But, I managed to complete the assignment.

The biggest complaint I have against my code is that I need a better way to use labeling. Some labels have to be in their certain places, since the programming is so huge, and labels had to be within -256 to 255 of the caller. It was a trick to set all the labels in locations that weren't in the middle of code, and didn't trash the layout of the code. Some labeling

refinements would be something I would refine before attempting a program again, and perhaps using labeling for branching. Linking probably solves this issue during compiling. My major improvements would be to optimize and organize the code more.