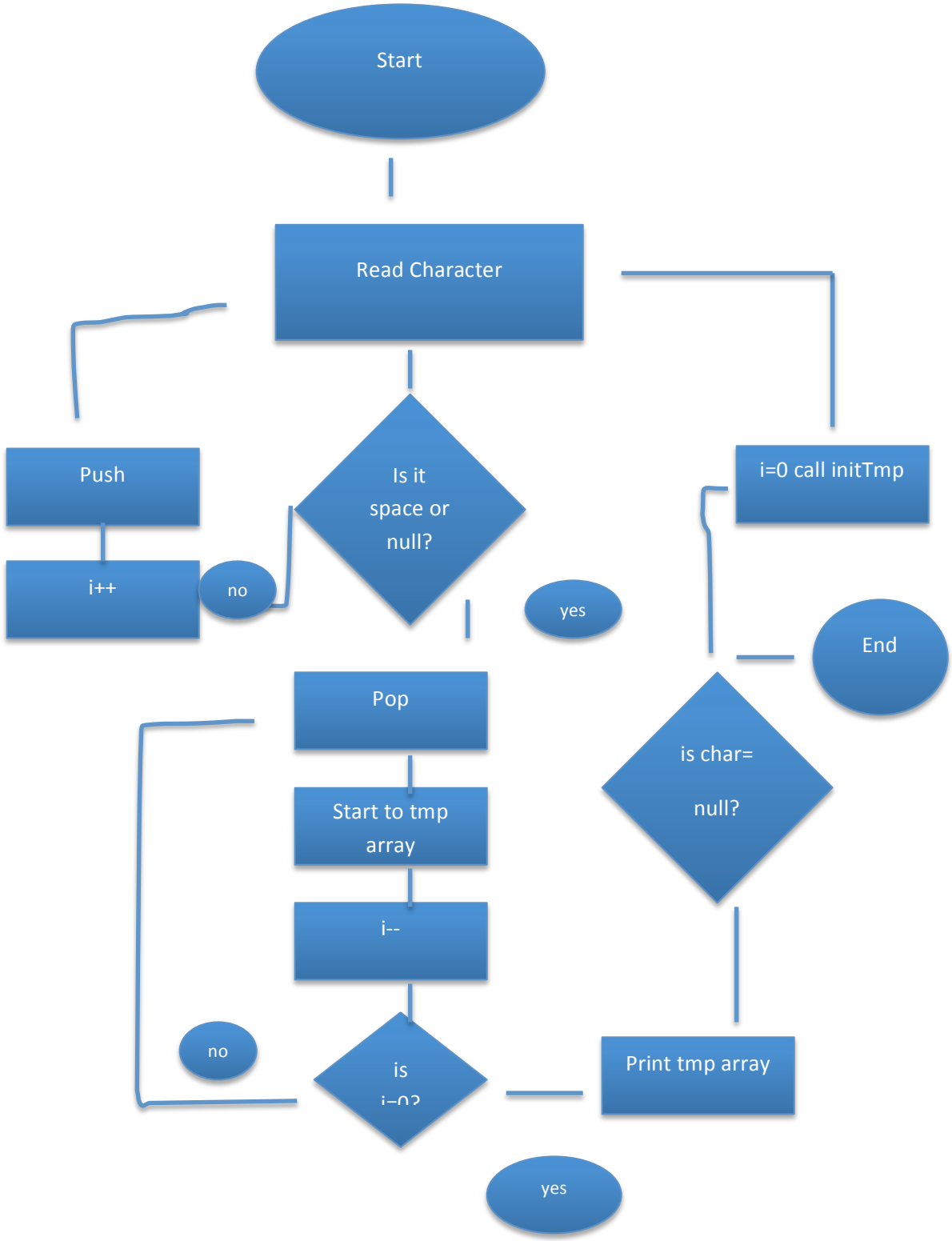
################################
Name: Priscilla Usmani- pusmani
Lab section: 3 MW TA: Jay Roldan
Due: 5/19/13
Lab Partner: Andrew Fisher- anmfishe
################################

Title:
Lab 5: Arduino/PIC32

Purpose:
The purpose of this lab is to create two assembly programs using MIPS assembly language. The first program creates a delay function that will accept an argument as number of milliseconds and then configure the light emitting diode on the Uno 32 board to turn on and off according to the duration of the delay subroutine. The second program uses stack to reverse the order of characters in a word. It takes the given string of characters then reverses them in the words but preserves the order of it.

Procedure:
In order to prep for the lab, my partner and I read through the entire lab, read the lecture notes on the PIC32 architecture, read about MIPS assembly, read through section 12.1-12.2, 3.3, and the table on page 75. This assignment was split into four parts. The first part was to practice compiling the "Hello,world" serial port example and testing it on TeraTerm. The second party is the instruction timing. We wrote a function with the label "mydelay". The function takes a single argument from register a0 in number of milliseconds to delay before returning. We used a serial print statement before and after a call to delay function, then convert the time into a number of loop iterations, then calculated the total number of instructions in the function and also the instructions per cycle, and then calibrate the "mydelay" by using at least 2 delay data points. The next part was the input/output section. We implemented a program that will blink LED 5 every 1 second. We used pin 58 on the microcontroller which is digital IO port F. In bit 0 we made it control the LED. We set all registers to perform different functions. We specifically only used TRISF register and PORTF register. PORTF was used to be able to ouput port by setting TRISF register bit 0 to a 0, then we wrote data bit 0 to PORTF setting the 0 to a 1 and turning the LED 5. The last part is the String Reversal. We used stack to reverse the string. It prints a string stored in memory. It reads each character of the string until it reaches a space or null character and if it isn't either one of them, it pushes it in the stack. If the character is a space or a null character, then it reverses the string by popping each character and storing it in a temporary array and printing it. If it is a null character, the program will end after printing the word. We used the starter code and then added more coding from the algorithms and other data listed below.

Algorithms and other data:



K

Pick-up Arduinos @ Bells

How to calculate delay:

Arduino: 80 Mhz

= 80 million cycles / sec

If you assume 1 instr / cycle, how many instr. to get 1 sec?

Loop:

li $a0 1000

jal mydelay

nop

j loop

nop

extra credit

- recognize double space
- recursive

push subroutine.

div. by 4 → addi $sp, $sp -4

sw $t1, 0($sp)

jr $sa

nop

mydelay:

const1 = k x a0

while const1 >= 0 {

const1 = const1 -1

while const1 >= 0 {

const = const1 -1

do nothing

}

}

5 cycle loop

- addi
- addi $t1, $t1, -1
- beq $t1 $0 Done
- nop
- j loop

string reversal:

labSb.s

ex: "Hello world"

output: Hello world

olleH

dlrow

msg [h|e|l|l|o|_|w|o|r|l|d]

push ↓ pointer ↑ stop return.

stack: [h|e|l|l|o]

pop (~~make~~ the pop)

temp: o l l e h

print temp (print: la $a1, temp) Jl - - -

What Went Wrong or What Were The Challenges:

The challenge in the program was part two. Getting the instruction timing was very difficult to cooperate with because the loops weren't responding the way that it should. After going to section and getting extra help we were able to convert the time into a number of lop iterations. The trick was to use a loop within a loop, but we ended up using only one loop.

Other Information:

When you configured PORTF bit 0 to be an output port you wrote a 1 on bit 0 of TRISF clear register. If we want to configure PORTF bits 3 and 4 to be an input port, what do we need to do? Include the a code that will configure PORTF.

You have to change the mask to x0002 or x003 depending on the bit you want to set it to. for bit 3 it would be x0004.

What did you have to do to calibrate your delay function? How did you calculate the number of times you have to loop?

I counted the number of instructions in my loop, 5. In order to get 80 we multiplied 16 by 5 times so we used 16000000.

What happens if you put a serial output in your delay function/loop? How does this impact the delay?

It will make the program longer since the serial takes a lot more cycles to output, but also depends on what you are outputting.

What happens if you forget to put a nop after your branch?

Nops are used to "slide" the code back into the correct placement after a branch/jump. The nop is used to align instruction addresses. If we forget the nop, it runs the line immediately after and it jumps.

If we are reading a sequence of integers instead of a string of characters, what necessary changes do you need in your program to make it work?

All variables we used to store a list on ints would have to change. Also, in any place we made an array of empty char spaces would have to change into int spaces. The store and load would also have to change and we'd have to use different commands.

Conclusion:

In conclusion, my partner and I have created two assembly programs using MIPS assembly language, one of which creates a delay function that will accept and argument as number of milliseconds, and configure the LED board to turn on and off according to the duration of the delay subroutine, and the other two use stack to reverse the order of characters in a word. The myDelay subrouting properly turns off and on and delays a 1000ms accurately, has the function that calibrated for other delays. The string reversal can easily recompile to display different strings, read characters from the string and recognize the space character and stacks correctly.