Name: Yazan Halawa

ECEn 425 HW 3

Instructor: Dr. Archibald

1. No this does NOT work. Assume an interrupt happens after the iHours variable is copied to iHoursTemp and while the calculation for the new time zone is being done. Let’s say that interrupt changes the value of hours. Then the calculation for iHoursTemp will be based on the wrong value and by the time the new value is returned we would have a totally incorrect result. For example let’s say the time was 3:59:59. So iHoursTemp is 3. Then an interrupt happen and the time is 4:00:00. The calculation for the new time zone settled on the value of (3-1)=2 for iHoursTemp and then that value is copied to iHours. So now the time is 2:00:00 when it should be 3:00:00. This is just one example of a totally off result.
2. If the microprossor has 16-bit registers: The result has to be stored in two registers, one for the higher bits and one for the lower bits. If an interrupt happens after reading the lower bits and before reading the higher bits, then the result could be as far off as 2^16 = 65536 bits representing the maximum amount of error for the upper bits(1 register of 16 bits = 1\*16 = 16).

If the microprocessor has 8-bit registers: The result has to be stored in 4 registers. The maximum amount of error would be if an interrupt happened after reading the first register and before reading the other three. The result could be as far off as 2^24 = 1.67E^7 bits (three registers each of 8 bits = 3\*8=24)

1. If the higher priority interrupt that uses ISecondsToday happens before the comparison in vUpdateTime then the time might not roll over correctly and cause incorrect results. Let’s say for example after incrementing in vUpdateTime, ISecondsToday = (60\*60\*24). Then right before the comparison the higher priority interrupt happens and sets ISecondsToday to (60\*60\*24)+1. So now it is higher than the maximum time and does not roll over. This way the till will keep increasing and will never roll over and the system fails. To fix this, I would disable interrupts inside vUpdateTime right at the beginning and enable them at the end. That way the ISecondsToday will not be modified before comparison.
2. a. Here is the code:

CPU 8086

ALIGN 2

Jmp main

ALIGN 2

main:

push bp

mov sp, bp

mov ax, 1

mov dx, 2

add sp, 2

push ax

add sp, 2

push dx

call myFunction

pop ax

pop dx

mov bp, sp

pop bp

b. argWord: word[bp+4]

argByte: byte[bp+6]

localWord: word[bp-2]

localByte: byte[bp-4]

c. The space for the local variables is allocated in the stack by just being pushed to it and moving the SP pointer to accommodate the new space and they are referenced using the BP pointer

1. A makefile is simply a way of associating short names, called targets, with a series of commands to execute when the action is requested. In the sample makefile, we are using the cc compiler with debug information turned on(-g) and using HP tools(-Aa). When the command make is run, the following happens:

Make(target){

If (target not specified)

target = the first target;

// Target = prog1

If (target has dependencies){// Which it does

// The following two lines create the three object files src1.o src2.o src3.o

For (each dependency)

Make (dependency);

// Now the linker takes the object files and links them to one executable

Run the commands specified for target;

}

else{

// Check if the files need updating

if (target is older than its dependencies)

Update the dependency files;

// Compile

Run the commands specified for target;

}

}

//Once the process is done: You should have a prog1 executable.