;Stephen Willson

;Program 3 – Due November 3, 2010

;This program will let the user type a key and will print out

;both the character and its corresponding ASCII value on the screen.

.model tiny

.stack

.code

.startup

; \*\*\*\*\* put program instructions here \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

mov cl,04h ;If you want to shift more than one place, ;you need to store the number of places you

;want to shift in CL and then use it as the ;source. Here, we put 04h into CL so later

;we can shift by 4, essentially isolating

;a single digit to be able to print

;it to the screen.

Begin:

mov ah,01h ;DOS interrupt 21h works by looking at the

int 21h ;value in AH and acts accordingly. 01h waits ;for key to be pressed and loads its ASCII ;value into AL. Then it prints the char to ;screen.

mov ch,al ;Stores the ASCII value of the key that was

mov ah,02h ;pressed in CH so it’s not overwritten.

mov dl,3Dh ;When int 21h sees 02 in AH, it will print

int 21h ;the ASCII value of whatever’s in DL. In ;this case, we put a 3D in DL because 3D is

;the ASCII code for ‘=’.

mov al,ch ;Puts ASCII for pressed key back into AL.

shr al,cl ;Here, we shift by 4 to the right, moving ;the most significant hex digit to the ;right. This lets us focus on just the most ;significant hex digit for now until it’s ;printed to the screen.

add al,30h ;We add 30h because 30h is the ASCII code ;for a zero. So, for example, if they type a ;4, the ASCII for a 4 is 34h. 04 + 30 = 34, ;the ASCII value for a 4.

cmp al,3Ah ;Compares to see if hex value in AL is

jc RST ;greater than ASCII value of 9 (39h). If it

add al,07h ;is bigger, the carry flag will not be set ;and the jump is skipped, meaning we add 07h ;for difference between numerical characters ;(0-9)and characters with ASCII codes higher ;than that. If not, the jump with carry ;fires and we skip ahead to the RST section,

;not adding the 07h and leaving the digit

;with just the 30h added.

RST:

mov dl,al ;Puts the ASCII value of the most

mov ah,02h ;significant hex digit into DL, which is

int 21h ;printed to the screen.

mov al,ch ;Puts ASCII value for the key pressed

and al,0Fh ;into AL. Then, we ‘and’ the 2 hex digits

;with 0Fh so the most significant digit

;becomes a 0 and the least significant digit ;remains the same. This way, we can mess ;with a single digit at a time.

add al,30h ;This addition is done for the same reason

;as the add above. We add to get the correct

;ASCII value for the typed key.

cmp al,3Ah ;Same comparison as above when we looked at

jc ND ;the most significant hex digit. The jump

add al,07h ;with carry and add commands will also fire ;under the same circumstances. The jump ;still will make the add line not fire, only ;this time, we jump to the section labeled ;ND.

ND:

mov dl,al ;Puts the ASCII value of the least

mov ah,02h ;significant hex digit in DL, which is

int 21h ;then printed to the screen.

mov dl,0Dh ;Then we do a line feed and a carriage

int 21h ;return to jump down to the next line in the

mov dl,0Ah ;command prompt where all the pressed keys

int 21h ;and their ASCII values are being printed.

mov al,ch ;Lastly, the ASCII value for the pressed

cmp al,40h ;key is stuck in AL. It’s compared to 40h,

jnz Begin ;the ASCII value for the @ sign. If the zero

;flag is set after the compare, that means

;the pressed key was @, and we skip the

;jump, ending the program. If the zero flag

;isn’t set, then they pressed some other key

;and the jump line fires, going back to the

;section of code labeled ‘Begin’, starting

;the whole process over.

; \*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*

mov ax ,4c00h ;quit

int 21h

end