THE MICROPROCESSORS & MICROCONTROLLERS

Instructor : The Tung Than
Student: Bach Hoang Phan Thanh
ID: 21520599

PRATICE EXERCISE #5:
ADDITION OF TWO 32-BIT NUMBERS ON THE 8086 PROCESSOR

1. Describe how sample code works:

* Sample code to add/subtract 2 number

```
name "add-sub"

org 100h

mov al, 5        ; bin=00000101b
mov bl, 10       ; hex=0ah or bin=00001010b

; 5 + 10 = 15 (decimal) or hex=0fh or bin=00001111b
add bl, al

; 15 - 1 = 14 (decimal) or hex=0eh or bin=00001110b
sub bl, 1

; print result in binary:
mov cx, 8
print: mov ah, 2    ; print function.
       mov dl, '0'
       test bl, 10000000b  ; test first bit.
       jz zero
       mov dl, '1'
zero:  int 21h
       shl bl, 1
loop print

; print binary suffix:
mov dl, 'b'
int 21h

; wait for any key press:
mov ah, 0
int 16h

ret
```

- Some instructions in the example code:
  + Mov: Copy operand2 to operand1.
  + Add: operand1 = operand1 + operand2
  + Sub: operand1 = operand1 - operand2
  + Test: Logical AND between all bits of two operands for flags only. These flags are effected: ZF, SF, PF. Result is not stored anywhere.
- The Print code:

```
print: mov ah, 2    ; print function.
       mov dl, '0'
       test bl, 10000000b  ; test first bit.
       jz zero
       mov dl, '1'
zero:  int 21h
       shl bl, 1
loop print
```

+ **mov ah, 2**: This line moves the value 2 into the ah register. In the context of the INT 21h interrupt in line 7, the value 2 in ah indicates the "print character" function of the interrupt.

+ **mov dl, '0'**: This line moves the ASCII value of the character '0' (48 in decimal) into the dl register. It sets the initial value to print before checking the bits of bl.

+ **test bl, 10000000b**: This line performs a bitwise AND operation between the value in the bl register and the binary value 10000000 (128 in decimal). It tests the first bit of the value in bl to determine if it is set or not.

+ **jz zero**: This line is a conditional jump instruction. If the zero flag is set (meaning the result of the previous test instruction was zero), the program jumps to the label zero, skipping the subsequent mov dl, '1' instruction. Otherwise, it continues execution to the next line.

+ **mov dl, '1'**: This line moves the ASCII value of the character '1' (49 in decimal) into the dl register. It sets the value to print when the first bit of bl is set.

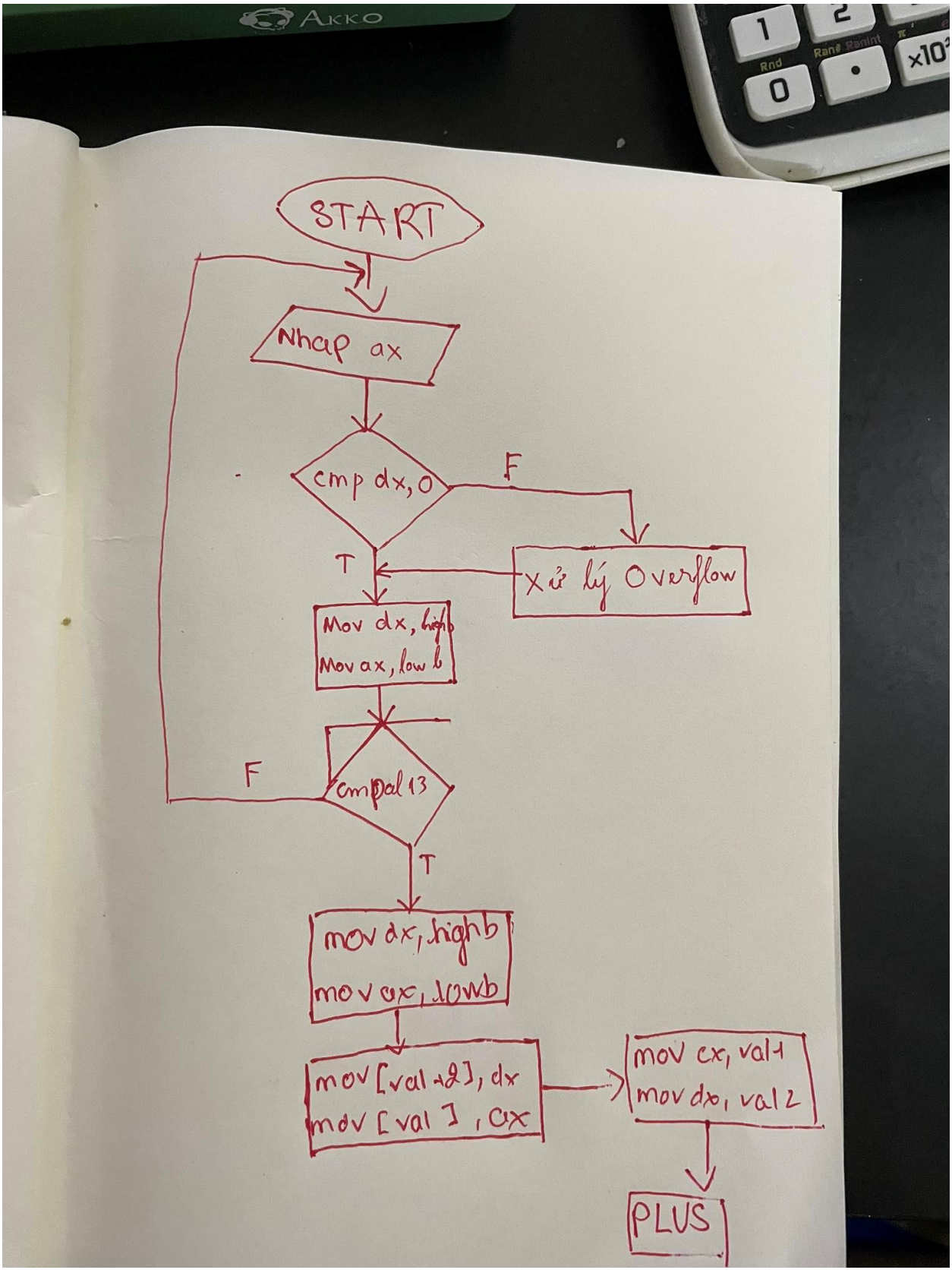
+ **zero: int 21h**: This line invokes the software interrupt 21h, which is a general-purpose interrupt in DOS/Windows that provides various services. In this case, the value in dl (either '0' or '1') will be printed to the console.

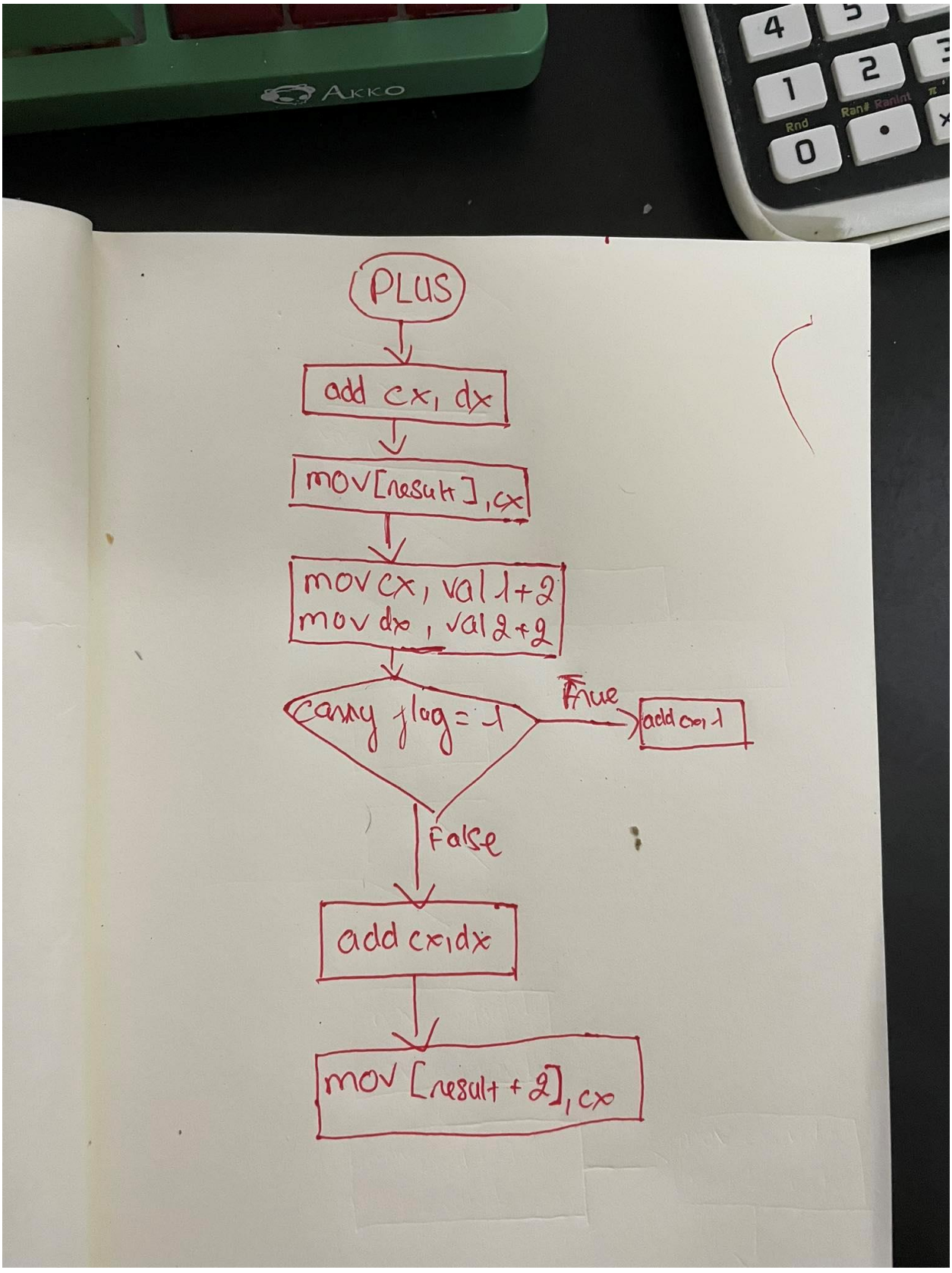
+ **shl bl, 1**: This line performs a logical shift left operation on the bl register by one bit. This effectively multiplies the value in bl by 2, shifting all bits one position to the left. This prepares bl for the next bit to be tested and printed.

+ **loop print:** This line decrements the value in the loop counter register (usually cx), and if the result is not zero, it jumps back to the print label. It creates a loop, allowing the printing process to repeat until all bits in bl have been tested and printed.

2. Flowchart of the program algorithm to add two 32-bit numbers

Flowchart:


START
Nhap ax
cmp dx, 0
F
T
Xử lý Overflow
Mov dx, high b
Mov ax, low b
cmpal 13
F
T
mov dx, high b
mov cx, low b
mov [val+2], dx
mov [val], cx
mov cx, val+1
mov dx, val2
PLUS


PLUS
add cx, dx
mov [result], cx
mov cx, val1+2
mov dx, val2+2
carry flag = 1
True
add cx, 1
False
add cx, dx
mov [result + 2], cx

3. Explain how the algorithm works, accompanied by a video (send a Google Drive link) to demonstrate the circuit operation in case the instructor cannot run the design file.

Link Result:
https://drive.google.com/drive/folders/1eVBvS7tUmC7ASB92y2F4T9ZVbr9OYTtu?usp=sharing

Code	Explanation
.model small	Specify the memory model as small (<64KB)
.stack 100h	Allocate 256 bytes of stack space
.data	
	Define String
tb1 db "Num1: $" tb2 db 10,13,"Num2: $" tb3 db 10,13,"Sum: $" tb4 db 10,13,"Difference: $" tb5 db 10,13,"Sum in 32-bit binary: $" tb6 db 10,13,"Difference in 32-bit binary: $"	$ = the end of the string
val1 dd ?, '$' val2 dd ?, '$' result dd ?, '$'	Define a 32-bit variable
x dd ?,'$' y dd ?,'$'	
highb dw ?,'$' lowb dw ?,'$'	Define a 16-bit variable to contain low, high bit

.code
main proc
mov ax,@data
mov ds,ax	

	Read Num1
mov ah,9	Print String
lea dx, tb1	Load the offset of tb1 string into DX
int 21h	Call DOS interupt to print string
call readNumber	
mov dx, highb	
mov ax, lowb	
mov [val1+2],dx	Store high word of the number
mov [val1],ax	Store low word of the number
;read Num2	
mov ah,9	
lea dx, tb2	
int 21h	
call readNumber	
mov dx, highb	
mov ax, lowb	
mov [val2+2],dx	
mov [val2],ax	
;plus Num1 and Num2	
mov ah,9	
lea dx, tb3	
int 21h	
call plus_Func	
mov si,word ptr result+2	Store high word into SI, use for printNumber
mov di,word ptr result	Store low word into DI, use for printNumber
call printNumber	
mov ah,9	
lea dx, tb5	
int 21h	
call printNumber32	

	Sub Num1 and Num2
mov ah,9 lea dx, tb4 int 21h call sub_Func	
mov si,word ptr result+2	Store high word into SI, use for printNumber
mov di,word ptr result	Store low word into DI, use for printNumber
call printNumber	
mov ah,9 lea dx, tb6 int 21h call printNumber32	
mov ah,4ch int 21h	Moves the immediate value 4Ch into the ah register. In the context of DOS interrupts, the value in the ah register specifies the function or service to be executed when invoking the interrupt. When this interrupt is triggered, the program terminates and control returns to the operating system. Stop the program
main endp	
readNumber proc xor dx,dx	
xor ax,ax	Clear ax, dx Set to 0
mov x,0 mov y,0	
mov bx,16	Read 16-bit
read: mov ah,1	Read Number

int 21h	
cmp al,13	Check if enter is PRESSED
je readDone	
cmp al,65	65 = 'A', check if the input is character or not
jle readNum	jle = jump if less than, if input < 65 => it is number
jmp readChar	If not, it is character
readNum:	
sub al, 48	48 = 0, to print Number Example, al = 1 (49 in ASCII) and have to sub 48 to equal to 1
jmp save_Number	
readChar:	
sub al, 55	55 = 7, to print Character Example, al = A(65 in ASCII) and have to sub 55 to equal to 10
save_Number:	
xor ah,ah	Clear ah
cmp dx,0	Check overflow-bit
jne ovf	
mov y,ax	
mov ax,x	
mul bx	Mul 16 to make 0 at the last-left bit
add ax,y	
mov x,ax	Store input number in x
mov highb,dx	Store high word into highb
mov lowb,ax	Store low word into lowb
jmp read	
ovf:	Overflow occurs (exceed 16bit number)
mov x,ax	
mov cx,4	Input is HEX, must shift left 4 bit Example 000A -> 00A0 <=> 0000 0000 0000 1010 -> 0000 0000 1010 0000
mov dx,highb	
mov ax,lowb	
shift_Bit:	

```
        shl ax,1                    Shift left AX
        rcl dx,1                    Rotate left DX with bit carry
        loop shift_Bit
        add ax,x
        mov highb,dx
        mov lowb,ax
        jmp read
    readDone:
    ret
readNumber  endp

printNumber  proc
    mov cx,4                        Input is HEX, must to shift left 4 bit
    mov bx,si
    print_High_Bit:
        xor dx,dx
        mov ax,4
        shift_Bit1:
        shl bx,1                    Shift left bx 1 bit
        rcl dx,1                    Rotate left dx with carry bit
        dec ax
        cmp ax,0                    CHECK LOOP
        jne shift_Bit1
        cmp dx,0ah                  Check dx is num
        jge printChar1
        add dx,30h                  Move DX to HEX string to print
        jmp print1
        printChar1:
        add dx,37h                  Move DX to HEX string to print

        print1:
        mov ah,2
        int 21h
    loop print_High_Bit

    mov cx,4
    mov bx,di
    print_Low_Bit:
        xor dx,dx
        mov ax,4
```

```
        shift_Bit2:
        shl bx,1
        rcl dx,1
        dec ax
        cmp ax,0
        jne shift_Bit2
        cmp dx,0ah
        jge printChar2
           add dx,30h
           jmp print2

        printChar2:
           add dx,37h

        print2:
        mov ah,2
           int 21h
    loop print_Low_Bit
    ret
printNumber endp



printNumber32 proc
    mov bx, word ptr result+2     Load the high word of the sum into BX

   mov cx, 16                     Loop counter, to print 16-bit register

printNumber32Loop:
    mov dl, '0'                   Default character is '0'
    test bx, 8000h                Check the leftmost bit
    jz printNumber32SkipBit       If it's 0, skip the next line
    mov dl, '1'                   Set the character to '1'

printNumber32SkipBit:
    mov ah, 2                     Print character function
    int 21h                       Print the character

    shl bx, 1                     Shift left to process the next bit
    loop printNumber32Loop        Loop until all bits are processed
```

mov bx, word ptr result	Load the low word of the sum into BX
mov cx, 16	Loop counter
printNumber32LoopLow:	
mov dl, '0'	
test bx, 8000h	
jz printNumber32SkipBitLow	
mov dl, '1'	
printNumber32SkipBitLow:	
mov ah,	
int 21h	
shl bx, 1	
loop printNumber32LoopLow	
ret	
printNumber32 endp	
plus_Func proc	
mov cx,word ptr val1	Load low word of val1 into CX
mov dx,word ptr val2	Load low word of val2 into DX
add cx,dx	
mov [result],cx	
mov cx,word ptr val1+2	
mov dx,word ptr val2+2	
jc carry1	If carry flag is set, jump to carry1
add cx,dx	
jmp save_Result1	
carry1:	
add cx, 1	Plus cx with 1 (1 is the bit-value of carry flag)
add cx, dx	
save_Result1:	
mov [result+2],cx	
ret	

plus_Func endp	
sub_Func proc	
mov cx,word ptr val1	
mov dx,word ptr val2	
sub cx,dx	
mov [result],cx	
mov cx,word ptr val1+2	
mov dx,word ptr val2+2	
jc carry2	If carry flag is set, jump to carry2
sub cx,dx	
jmp save_Result2	
carry2:	
sub cx,1	Sub cx with 1 <1 is the bit-vaule of carry flag)
sub cx,dx	
save_Result2:	
mov [result+2],cx	
ret	
sub_Func endp	
END	