; Juan Pedraza

; CSCI 112 – ProjectPhase1

;Program will simulate a 1 bit ALU

;Will as for 4 inputs Ai, Bi, Ci, Op

; Will display inputs along with results Si, Co

; Op= 00->AND , 01->OR, 10->ADD

.586

.MODEL FLAT

INCLUDE io.h

.STACK 4096

.DATA

askInput BYTE "Enter ai, bi, ci, op. (space separated)",0

inputStr BYTE 11 DUP(?),0

inputArr DWORD 4 DUP(0) ;array to store value of inputs

varSi DWORD 0

varCo DWORD 0

count1 DWORD 0

results BYTE "Results:",0

finalTitle BYTE " Ai Bi Ci Op Si Co",0 ;63 char

finalStr BYTE 240 DUP(?),0

temp BYTE 11 DUP(?),0

tempx BYTE 11 DUP(" "),0

.CODE

\_MainProc PROC

input askInput, inputStr, 11

lea ebx, inputStr ;get address of inputStr

outerLoop: inc count1 ;increase loop counter

lea esi, tempx ;flush temp str before using

lea edi, temp

cld

mov ecx, 11

rep movsb

lea edx, temp

innerLoop: cmp BYTE PTR[ebx], 20H ;if ending mark (space)

je done1

mov AL, BYTE PTR[ebx] ;otherwise get 1 byte from input string

mov [edx], AL ;move to temp

inc ebx ;next byte in input string

inc edx ;next byte in tmp string

jmp innerLoop

done1: atod temp

mov ecx, count1

dec ecx ;index starts at 0

imul ecx, 4 ;array element size = 4 bytes

mov inputArr[ecx], eax ;store one value in array

inc ebx ;skip space char

cmp count1, 4 ;loop 4 times for each input value

jnge outerLoop

;call ALU with arguments

;push inputArr[12] ;Op

;push inputArr[8] ;Ci

;push inputArr[4] ;Bi

;push inputArr[0] ;Ai

;call ALUfunc

;add esp, 16

lea eax, inputArr ;send array

push eax

call ALUfunc

add esp, 4

mov varSi, eax ;eax has the Si output of the ALU function

mov varCo, ebx ;ebx has the Co output of the ALU function (carry out)

;generate final string

lea esi, finalTitle

lea edi, finalStr

mov ecx, 63

cld

rep movsb

mov finalStr[63], 0dH ;newline

lea edi, finalStr[64]

dtoa temp, inputArr[0] ;get first value (Ai)

lea esi, temp

mov ecx, 11

cld

rep movsb

lea edi, finalStr[64+11]

dtoa temp, inputArr[4] ;get 2nd value (Bi)

lea esi, temp

mov ecx, 11

cld

rep movsb

lea edi, finalStr[64+22]

dtoa temp, inputArr[8] ;get 3rd value (Ci)

lea esi, temp

mov ecx, 11

cld

rep movsb

lea edi, finalStr[64+33]

dtoa temp, inputArr[12] ;get 4th value (Op)

lea esi, temp

mov ecx, 11

cld

rep movsb

lea edi, finalStr[64+44]

dtoa temp, varSi

lea esi, temp

mov ecx, 11

cld

rep movsb

lea edi, finalStr[64+55]

dtoa temp, varCo

lea esi, temp

mov ecx, 11

rep movsb

output results, finalStr

ret

\_MainProc ENDP

; ALU Function calls AND (00), OR (01), Add (10) functions

; function takes in a reference to an array

; ALUfunc(array[])

; function returns Si in eax register and Co in ebx register

ALUfunc PROC

push ebp ;save base pointer

mov ebp, esp ;establish stack frame

push ecx

push edx

push esi

mov esi, DWORD PTR [ebp+8] ;get reference to array

;call ANDfunc

mov ecx, [esi]

push ecx

add esi, 4

mov edx, [esi]

push edx

sub esi, 4 ;reset array index back to begining

call ANDfunc

add esp, 8 ;results from ANDfunc in eax

;call ORfunc

push ecx ;Ai

push edx ;Bi

call ORfunc ;result stored in ebx

add esp, 8

;call ADDfunc

push ecx ;Ai

push edx ;Bi

add esi, 8 ;move array index to Ci

mov edx, [esi]

push edx ;Ci

sub esi, 8 ;reset index of array

call ADDfunc ;sum in ecx and Co in edx

add esp, 12

;after first 3 calls AND->eax, OR->ebx, ADD-> ecx & edx

;call MUXfunc(Op code, AND , OR, Si, Co, Ci)

add esi, 8

pushd DWORD PTR [esi] ;push Ci

sub esi, 8 ;reset array index

push edx ;push Co

push ecx ;push Si

push ebx ;OR result

push eax ;AND result

add esi, 12 ;move to op

pushd DWORD PTR [esi] ;push Op code onto stack

call MUXfunc ;results Si in eax, Co -> ebx

add esp, 24

pop esi

pop edx

pop ecx

pop ebp

ret

ALUfunc ENDP

; AND function

; takes 2 values as inputs Ai and Bi result is stored in eax

ANDfunc PROC

push ebp

mov ebp, esp

push ebx

mov eax, [ebp+8]

mov ebx, [ebp+12]

add eax, ebx

cmp eax, 2

jne false1

true1: mov eax, 1 ;set to true

jmp exit1

false1: mov eax, 0 ;set to false

exit1:

pop ebx

pop ebp

ret

ANDfunc ENDP

; OR function

; Takes 2 values as inputs Ai and Bi result stored in ebx

ORfunc PROC

push ebp

mov ebp, esp

push eax

mov eax, [ebp+8]

mov ebx, [ebp+12]

add ebx, eax

cmp ebx, 1

jl false2

true2: mov ebx, 1 ;true only one is

jmp exit2

false2: mov ebx, 0 ;set to false

exit2:

pop eax

pop ebp

ret

ORfunc ENDP

; Add function

; Takes in 3 inputs Ai, Bi, Ci results Si->ecx , Co->edx

ADDfunc PROC

push ebp

mov ebp, esp

push eax

push ebx

mov eax, [ebp+8] ;Ci

mov ebx, [ebp+12] ;Bi

mov ecx, [ebp+16] ;Ai

add ecx, ebx

add ecx, eax

cmp ecx, 3

je three1

cmp ecx, 2

je two1

cmp ecx, 1

je one1

;must be 0

mov ecx, 0

mov edx, 0

jmp exit3

three1: mov ecx, 1

mov edx, 1

jmp exit3

two1: mov ecx, 0

mov edx, 1

jmp exit3

one1: mov ecx, 1

mov edx, 0

jmp exit3

exit3:

pop ebx

pop eax

pop ebp

ret

ADDfunc ENDP

; Multiplexer function

;MUXfunc(Op code, AND , OR, Si, Co, Ci)

;oupts are Si->eax and Co->ebx

MUXfunc PROC

push ebp

mov ebp, esp

push ecx

mov ecx, [ebp+8] ;get op code

cmp ecx, 10 ;check if ADD op code

jne check2

mov eax, [ebp+20]

mov ebx, [ebp+24]

jmp exit4

check2: cmp ecx, 1 ;check if OR op code

jne check3

mov eax, [ebp+16]

mov ebx, [ebp+28]

jmp exit4

check3: cmp ecx, 0 ;don't need to check since it is the last option

jne exit4 ;incase i need to change later

mov eax, [ebp+12]

mov ebx, [ebp+28]

jmp exit4

exit4:

pop ecx

pop ebp

ret

MUXfunc ENDP

END

