

```

;Bee Cha
;Lab 9
;Implement a 1-bit ALU
;Create gate procedures
;Then perform 1-bit ALU instructions
;Output results

```

```

.586
.MODEL FLAT

```

```

INCLUDE io.h          ; header file for input/output

```

```

.STACK 4096

```

```

.DATA

```

```

prompt1    BYTE    "Enter a, b, carry in, and op code", 0
a          DWORD    ?
b          DWORD    ?
cin        DWORD    ?
op         DWORD    ?
arg        BYTE     40 dup(?),0
temp       BYTE     4 dup(" "),0
temp2      BYTE     ?
and_result DWORD    ?
or_result  DWORD    ?
xor_result DWORD    ?
add_result DWORD    ?
cin_plus   DWORD    ?
si_result  DWORD    ?
result_str BYTE     2 dup(" "), 0
outlbl1    BYTE     "Result 1-Bit ALU", 0
label2     BYTE     "ai", 09h, "bi", 09h, "ci", 09h, "op", 09h, "Si", 09h, "Ci+1", 0dh, 0      ;20
characters
outstr     BYTE     400 dup(?), 0

```

```

.CODE

```

```

_MainProc PROC

```

```

        input  prompt1, arg, 40          ;Get user input

```

```

        lea     ebx, arg
        lea     edx, temp
        mov     ecx, 0

```

```

loop1:

```

```

        cmp     BYTE ptr [ebx], 20h      ;Compare if it's a blank space
        je      done1
        cmp     BYTE ptr [ebx], 00h      ;Compare if it's a null space
        je      done1
        mov     AL, byte ptr [ebx]       ;Otherwise get 1 byte from it
        mov     [edx], AL                ;And put it in temp string
        inc     ecx
        inc     edx

```

```

done1:

```

```

        inc     ebx
        cmp     ecx, 4
        jl      loop1

```

;-----Convert each parameters in [EBX] into dword, ATOD-----;

```
    lea        edx, temp
    mov        AL, byte ptr [edx]
    mov        temp2, AL
    atod      temp2
    mov        a, eax

    inc        edx
    mov        AL, byte ptr [edx]
    mov        temp2, AL
    atod      temp2
    mov        b, eax

    inc        edx
    mov        AL, byte ptr [edx]
    mov        temp2, AL
    atod      temp2
    mov        cin, eax

    inc        edx
    mov        AL, byte ptr [edx]
    mov        temp2, AL
    atod      temp2
    mov        op, eax
```

;-----Now you have the corresponding arguments: a, b, cin, and op-----;

```
    push      a
    push      b
    call      AND_gate          ;results stored in eax
    mov       and_result, eax

    call      OR_gate
    mov       or_result, eax

    call      XOR_gate
    mov       xor_result, eax
    add       esp, 8

    push      and_result
    push      or_result
    push      cin

    call      full_adder
    mov       add_result, eax
    add       esp, 12
```

;-----A XOR B AND CIN OR A AND B-----;

;-----xor_result AND CIN OR and_result

```
    mov       eax, cin
    and       eax, xor_result
    or        eax, and_result
    mov       cin_plus, eax
```

;-----MULTIPLEXER-----;

```
;OP Code: 0      AND
;OP Code: 1      OR
;OP Code: 2      ADD
```

```

        cmp     op, 0
        je      AND_label
        cmp     op, 1
        je      OR_label
        cmp     op, 2
        je      ADD_label

```

AND_label:

```

        ;-----and_result AND or_result AND add_result-----;
        mov     eax, and_result
        and     eax, or_result
        and     eax, add_result
        mov     si_result, eax
        jmp     done2

```

OR_label:

```

        ;-----and_result OR or_result OR add_result-----;
        mov     eax, and_result
        or      eax, or_result
        or      eax, add_result
        mov     si_result, eax
        jmp     done2

```

ADD_label:

```

        ;-----and_result ADD or_result ADD add_result-----;
        mov     eax, and_result
        add     eax, or_result
        add     eax, add_result
        mov     si_result, eax
        jmp     done2

```

done2:

```

        ;-----Append string to output window-----;

```

```

        lea     esi, label2
        lea     edi, outstr
        cld
        mov     ecx, 20
        rep     movsb

        lea     esi, temp
        lea     edi, outstr+20
        cld
        mov     ecx, 1
        rep     movsb
        mov     outstr+21, 09h

        lea     esi, temp+1
        lea     edi, outstr+22
        mov     ecx, 1
        rep     movsb
        mov     outstr+23, 09h

        lea     esi, temp+2
        lea     edi, outstr+24
        mov     ecx, 1
        rep     movsb
        mov     outstr+25, 09h

```

```

        lea     esi, temp+3
        lea     edi, outstr+26
        mov     ecx, 1
        rep     movsb
        mov     outstr+27, 09h

        dtoa    temp2, si_result

        lea     esi, temp2+10
        lea     edi, outstr+28
        mov     ecx, 1
        rep     movsb
        mov     outstr+29, 09h

        dtoa    temp2, cin_plus

        lea     esi, temp2+10
        lea     edi, outstr+30
        mov     ecx, 1
        rep     movsb

        output  outlbl, outstr

        mov     eax, 0 ; exit with return code 0
        ret
_MainProc ENDP

```

;-----GATE PROCEDURES-----;

AND_gate PROC

;-----AND_gate takes parameters a and b then it will AND the results back-----;

```

        push    ebp
        mov     ebp, esp

        mov     eax, [ebp+12]
        and     eax, [ebp+8]

        pop     ebp
        ret

```

AND_gate ENDP

OR_gate PROC

;-----OR_gate takes parameters a and b then it will OR the results back-----;

```

        push    ebp
        mov     ebp, esp

        mov     eax, [ebp+12]
        or      eax, [ebp+8]

        pop     ebp
        ret

```

OR_gate ENDP

XOR_gate PROC

```
push    ebp
mov     ebp, esp
mov     eax, [ebp+12]
xor     eax, [ebp+8]

pop     ebp
ret
```

XOR_gate ENDP

full_adder PROC

```
;-----Full Adder: Ai XOR Bi XOR cin-----;
push    ebp
mov     ebp, esp

mov     eax, [ebp+8]      ;Ai
xor     eax, [ebp+12]     ;Ai XOR Bi
xor     eax, [ebp+16]     ;Ai XOR Bi XOR cin

pop     ebp
ret
```

full_adder ENDP

END ; end of source code

Result 1-Bit ALU					
ai	bi	ci	op	Si	Ci+1
0	0	0	0	0	0
0	1	1	0	0	1
1	0	0	0	0	0
1	1	1	0	1	1
0	0	1	1	1	0
0	1	0	1	1	0
1	0	1	1	1	1
1	1	0	1	1	1
0	0	0	2	0	0
0	0	1	2	1	0
0	1	0	2	2	0
0	1	1	2	1	1
1	0	0	2	2	0
1	0	1	2	1	1
1	1	0	2	2	1
1	1	1	2	3	1