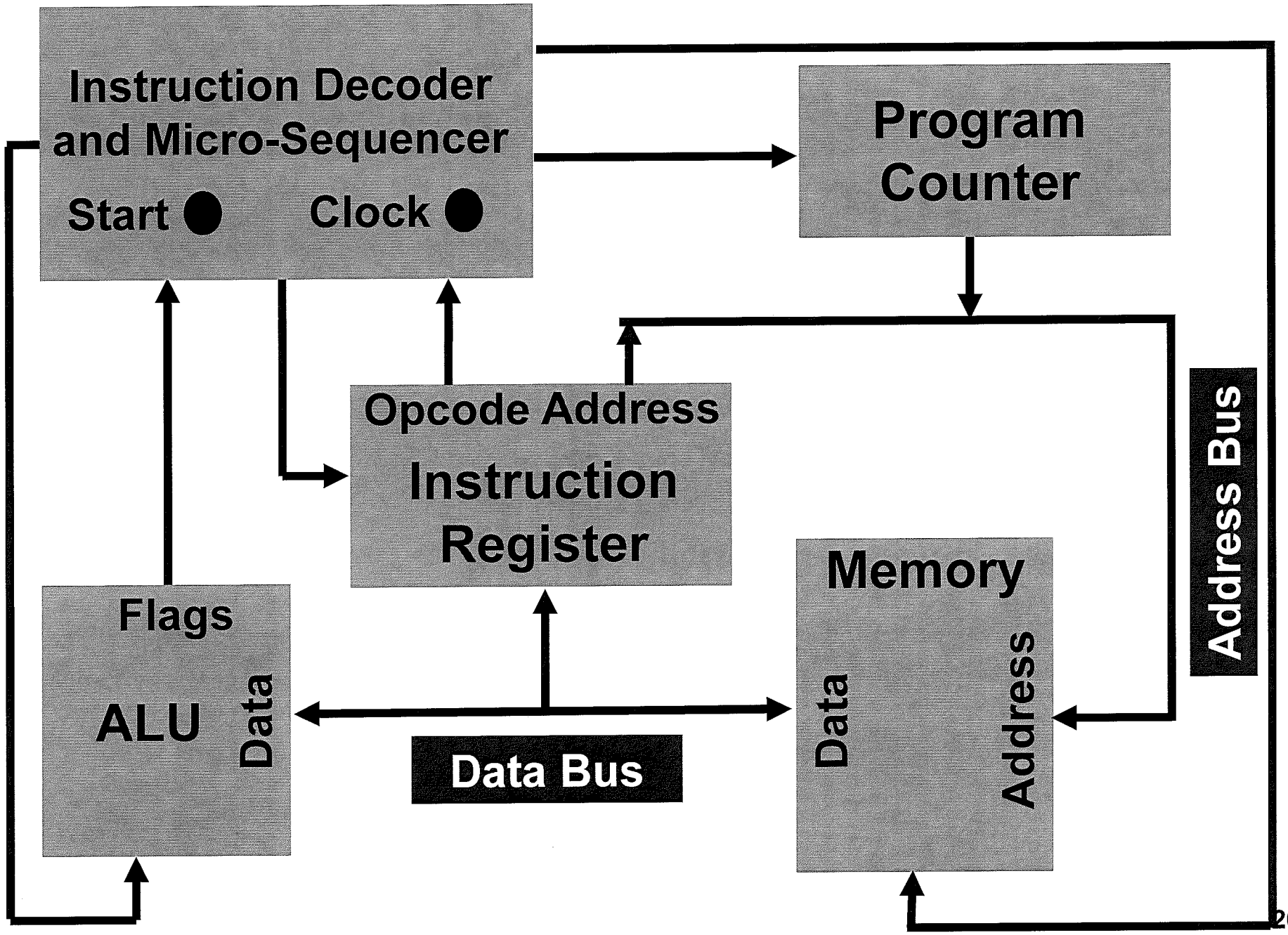

Instruction Decoder
and Micro-Sequencer
Start
Clock
Program
Counter
Opcode Address
Instruction
Register
Flags
ALU
Data
Data Bus
Memory
Data
Address
Address Bus

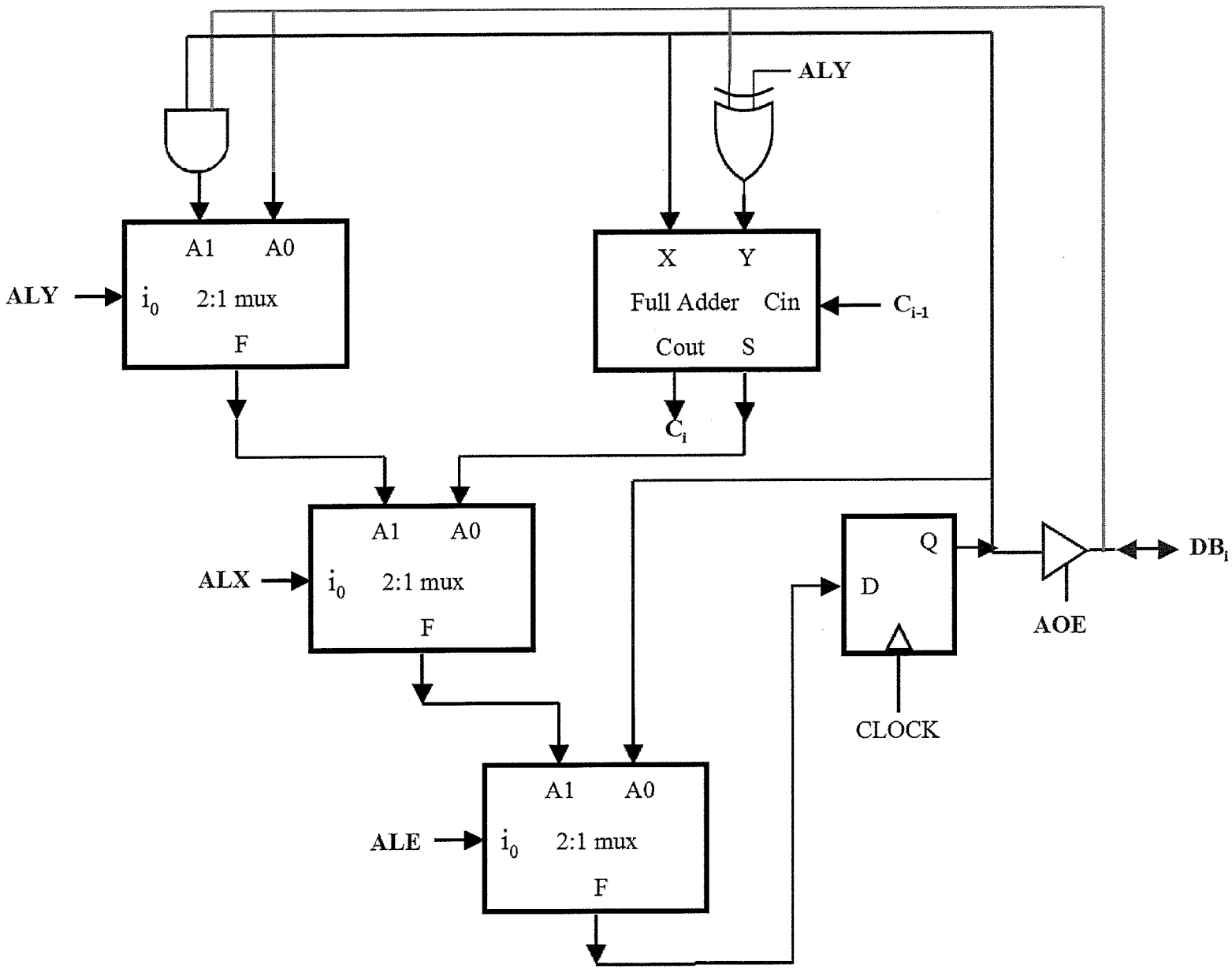
```
MODULE alu

TITLE 'ALU Module'

"   8-bit, 4-function ALU with bi-directional data bus
"
"   ADD:  (Q7..Q0) <- (Q7..Q0) + DB7..DB0
"   SUB:  (Q7..Q0) <- (Q7..Q0) - DB7..DB0
"   LDA:  (Q7..Q0) <-  DB7..DB0
"   AND:  (Q7..Q0) <- (Q7..Q0) & DB7..DB0
"   OUT:  Value in Q7..Q0 output on data bus DB7..DB0
"
"   AOE  ALE  ALX  ALY   Function    CF  ZF  NF  VF
"   ===  ===  ===  ===   ========    ==  ==  ==  ==
"    0    1    0    0    ADD         X   X   X   X
"    0    1    0    1    SUB         X   X   X   X
"    0    1    1    0    LDA         •   X   X   •
"    0    1    1    1    AND         •   X   X   •
"    1    0    d    d    OUT         •   •   •   •
"    0    0    d    d    <none>      •   •   •   •
"
"    X -> flag affected   • -> flag not affected
"
"  Note: If ALE = 0, the state of all register bits should be retained
```

```
DECLARATIONS

CLOCK pin;

" ALU control lines (enable & function select)

ALE pin; " overall ALU enable
AOE pin; " data bus tri-state output enable
ALX pin; " function select
ALY pin;

" Carry equations (declare as internal nodes)

CY0..CY7 node istype 'com';

" Combinational ALU outputs (D flip-flop inputs)
" Used for flag generation (declare as internal nodes)

ALU0..ALU7 node istype 'com';

" Bi-directional 8-bit data bus (also, accumulator register bits)

DB0..DB7 pin istype 'reg_d,buffer';

" Condition code register bits

CF pin istype 'reg_d,buffer';  " carry flag
VF pin istype 'reg_d,buffer';  " overflow flag
NF pin istype 'reg_d,buffer';  " negative flag
ZF pin istype 'reg_d,buffer';  " zero flag
```

```
" Declaration of intermediate equations

" Least significant bit carry-in (0 for ADD, 1 for SUB => ALY)

CIN = ALY;

" Intermediate equations for adder/subtractor SUM (S0..S7),
" selected when ALX = 0

S0 = DB0.q $ (DB0.pin $ ALY) $ CIN;
S1 = DB1.q $ (DB1.pin $ ALY) $ CY0;
S2 = DB2.q $ (DB2.pin $ ALY) $ CY1;
S3 = DB3.q $ (DB3.pin $ ALY) $ CY2;
S4 = DB4.q $ (DB4.pin $ ALY) $ CY3;
S5 = DB5.q $ (DB5.pin $ ALY) $ CY4;
S6 = DB6.q $ (DB6.pin $ ALY) $ CY5;
S7 = DB7.q $ (DB7.pin $ ALY) $ CY6;

" Intermediate equations for LOAD and AND, selected when ALX = 1

L0 = !ALY&DB0.pin # ALY&DB0.q&DB0.pin;
L1 = !ALY&DB1.pin # ALY&DB1.q&DB1.pin;
L2 = !ALY&DB2.pin # ALY&DB2.q&DB2.pin;
L3 = !ALY&DB3.pin # ALY&DB3.q&DB3.pin;
L4 = !ALY&DB4.pin # ALY&DB4.q&DB4.pin;
L5 = !ALY&DB5.pin # ALY&DB5.q&DB5.pin;
L6 = !ALY&DB6.pin # ALY&DB6.q&DB6.pin;
L7 = !ALY&DB7.pin # ALY&DB7.q&DB7.pin;
```

```
EQUATIONS

" Ripple carry equations (CY7 is COUT)
CY0 = DB0.q&(ALY$DB0.pin) # DB0.q&CIN # (ALY$DB0.pin)&CIN;
CY1 = DB1.q&(ALY$DB1.pin) # DB1.q&CY0 # (ALY$DB1.pin)&CY0;
CY2 = DB2.q&(ALY$DB2.pin) # DB2.q&CY1 # (ALY$DB2.pin)&CY1;
CY3 = DB3.q&(ALY$DB3.pin) # DB3.q&CY2 # (ALY$DB3.pin)&CY2;
CY4 = DB4.q&(ALY$DB4.pin) # DB4.q&CY3 # (ALY$DB4.pin)&CY3;
CY5 = DB5.q&(ALY$DB5.pin) # DB5.q&CY4 # (ALY$DB5.pin)&CY4;
CY6 = DB6.q&(ALY$DB6.pin) # DB6.q&CY5 # (ALY$DB6.pin)&CY5;
CY7 = DB7.q&(ALY$DB7.pin) # DB7.q&CY6 # (ALY$DB7.pin)&CY6;

" Combinational ALU equations
ALU0 = !ALX&S0 # ALX&L0;
ALU1 = !ALX&S1 # ALX&L1;
ALU2 = !ALX&S2 # ALX&L2;
ALU3 = !ALX&S3 # ALX&L3;
ALU4 = !ALX&S4 # ALX&L4;
ALU5 = !ALX&S5 # ALX&L5;
ALU6 = !ALX&S6 # ALX&L6;
ALU7 = !ALX&S7 # ALX&L7;

" Register bit and data bus control equations
[DB0..DB7].d = !ALE&[DB0..DB7].q # ALE&[ALU0..ALU7];

[DB0..DB7].clk = CLOCK;

[DB0..DB7].oe = AOE;
```

ALU Multiplexer Block Diagram



```
" Flag register state equations

CF.d = !ALE&CF.q # ALE&(!ALX&(CY7 $ ALY) # ALX&CF.q);

CF.clk = CLOCK;

ZF.d = !ALE&ZF.q # ALE&(!ALU7&!ALU6&!ALU5&!ALU4&!ALU3&!ALU2&!ALU1&!ALU0);

ZF.clk = CLOCK;

NF.d = !ALE&NF.q # ALE&ALU7;

NF.clk = CLOCK;

VF.d = !ALE&VF.q # ALE&(!ALX&(CY7 $ CY6) # ALX&VF.q);

VF.clk = CLOCK;

END
```