Tutorial

20/11/12

Identify function of the given set of instructions

```
INTS    PROC    FAR  USES   AX
        ADD  AX,BX
        ADD  AX,BP
        ADD  AX,DI
        ADD  AX,SI
        IRET
INTS    ENDP
```

Software Interrupt

- Whenever a software interrupt instruction executes, it
  - Pushes the flags onto the stack
  - Clears the T & I flag bits
  - Pushes CS on the stack
  - Fetches the new value for CS from the interrupt vector
  - Pushes IP onto the stack
  - Fetches the new value for IP from the vector
  - Jumps to the new location addressed by CS & IP

INT 10 H	Video Services
INT 11H	Equipment List
INT 12H	Usable memory size
INT 13H	Disk I/O
INT 14H	Serial Port I/O
INT 15H	AT service
INT 16H	Keyboard I/O
INT 17H	PrinterI/O
INT 18H	ROM-BASIC
INT 19H	Bootstrap
INT 1aH	Time I/O
INT 1bH	Keyboard Break
INT 1cH	User timer Interrupt
INT 20-2fH	DOS system calls
INT 67H	Expanded memory function

INT 21H,FUNCTION 4CH

- Terminates with return code-EXIT(n)
- USED TO TERMINATE THE PROGRAM WITH RETURN CODE.
- AH=4C
- AL=RETURN CODE
- RETURN CODE 0 : TERMINATION WITH SUCCESSFUL EXECUTION
- DOS sets the error level to the return code

INT 21H,Functionn 09H

- Display String
- AH=09H
- DS:DX=segment:offset of the string terminated by the symbol$
- Writes a string terminated with $ to the std. output device
- Pgm has to explicitly handle the disk space availability

Display a message on screen using MACROS

```
printstring                 macro       msg
            mov  ah,09h
            mov  dx,offset msg
            INT 21H
            endm
_DATA segment
      cr        equ         0dH
      lf        equ         0aH
      msg1 db  'HELLO WORLD', cr, lf, '$'
      msg2 db 'HOPE YOU ALL ARE FINE', '$'
_DATA ends
_CODE segment
      assume CS:_CODE, ds:_DATA
Start:      mov  ax,_DATA
            mov  ds,ax
            printstring msg1
            printstring msg2
            mov  ah,4cH
            mov  al,00H
            INT 21H
_CODE ends
            end start
```

Write comment against each line of following program

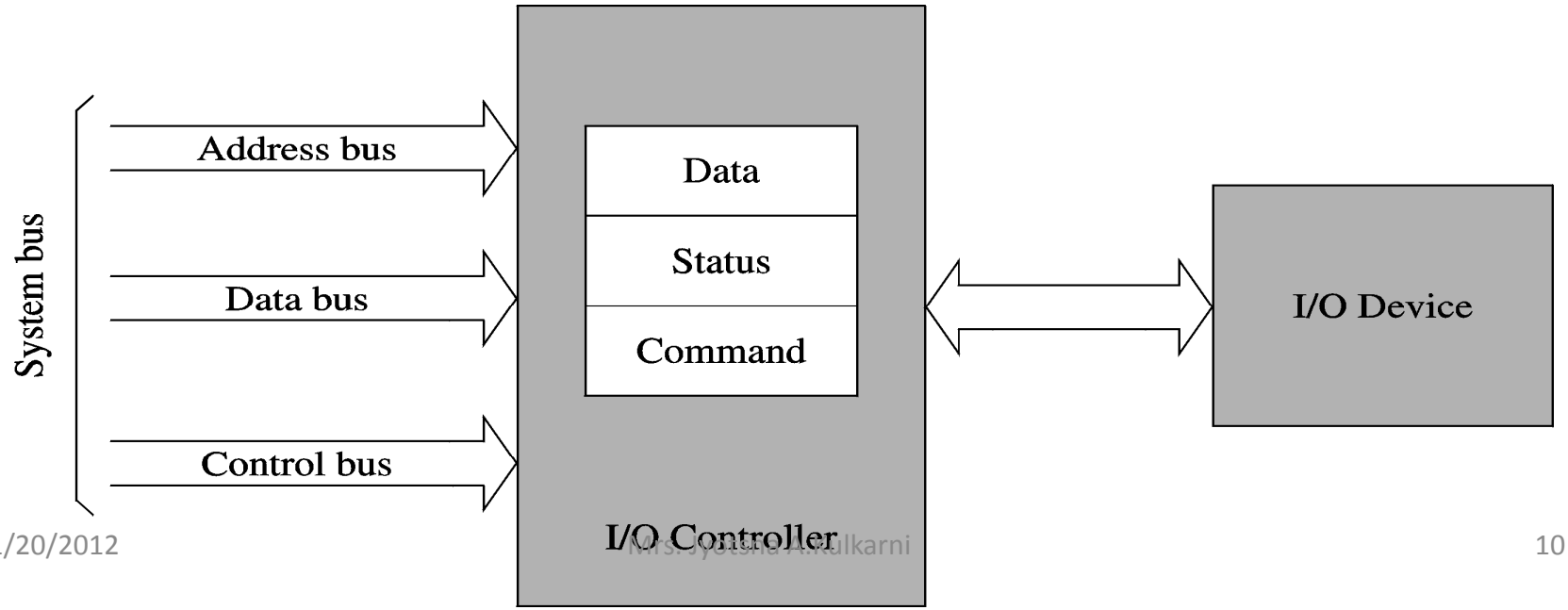
```
                    MOV AX,30
                    MOV BX,40
                    PUSH AX
                    PUSH BX
0008 E8 0066        CALL      ADDM

0071      ADDM      PROC      NEAR
                    PUSH BP
                    MOV BP,SP
                    MOV AX,[BP+4]
                    ADD AX, [BP+6]
                    POP BP
                    RET 4
ADDM                ENDP
```

Hardware Interrupts

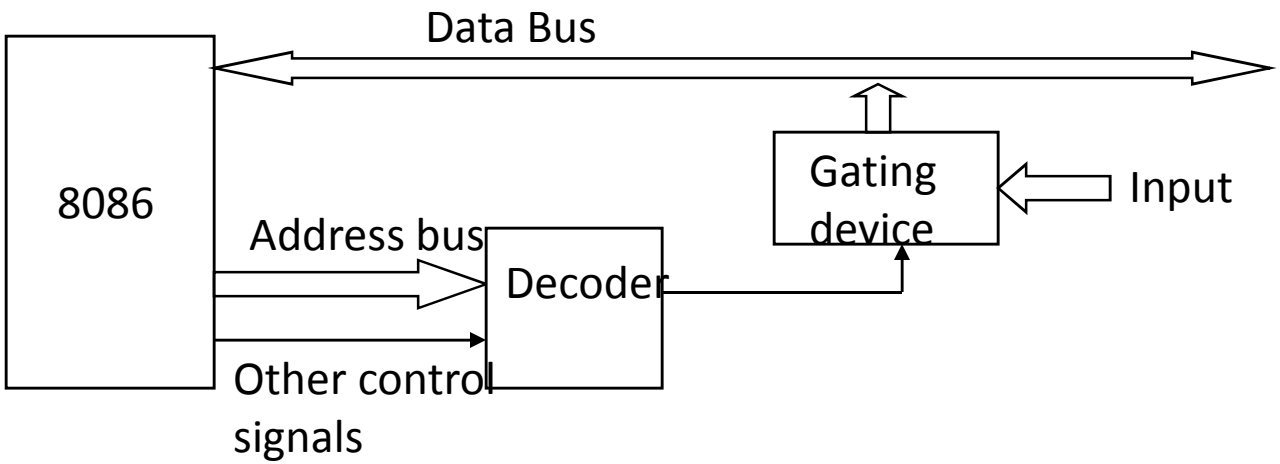
INT 00 H	Division by zero
INT 01H	Single step
INT 02H	Non-Maskable
INT 03H	Breakpoint
INT 04H	Overflow
INT 05H	Print Screen
INT 06H	Reserved
INT 07H	Reserved
INT 08H	Timer
INT 09H	Keyboard
INT 0a-0dH	Hardware interrupts
INT 0eH	Diskette
INT 0fH	Hardware interrupt

I/O device access

- To communicate with an I/O device, we need
  - Access to various registers (data, status,…)
    - This access depends on I/O mapping
      - Two basic ways
        - » Memory-mapped I/O
        - » Isolated I/O



Input Port Implementation



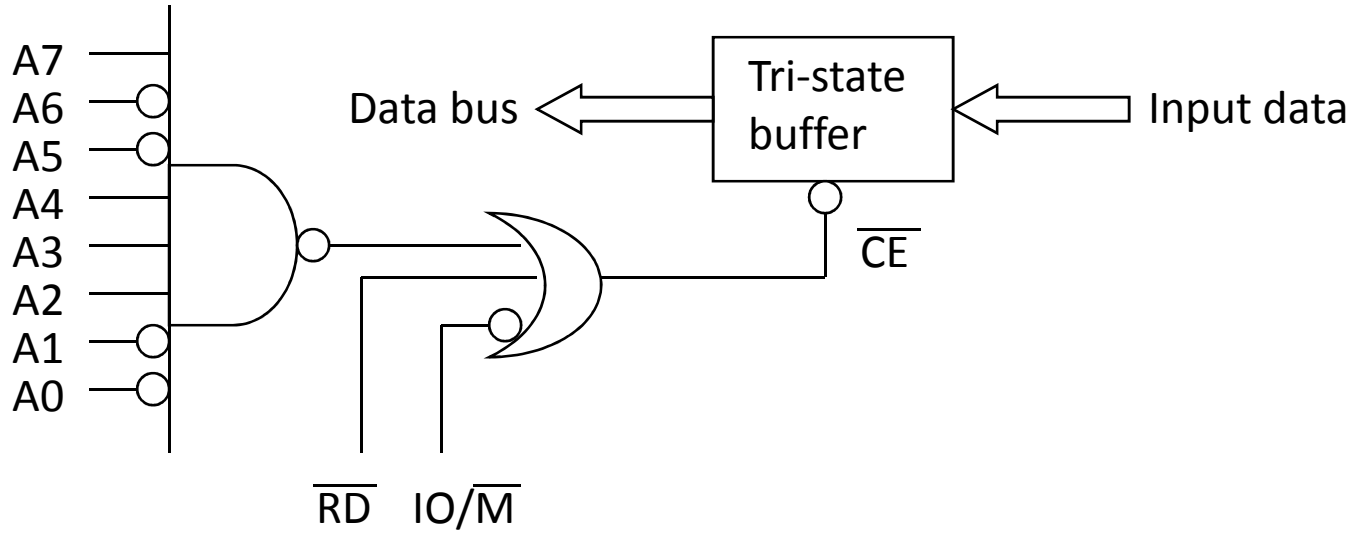
— The outputs of the gating device are high impedance when the processor is not accessing the input port

— When the processor is accessing the input port, the gating device transfers input data to CPU data bus

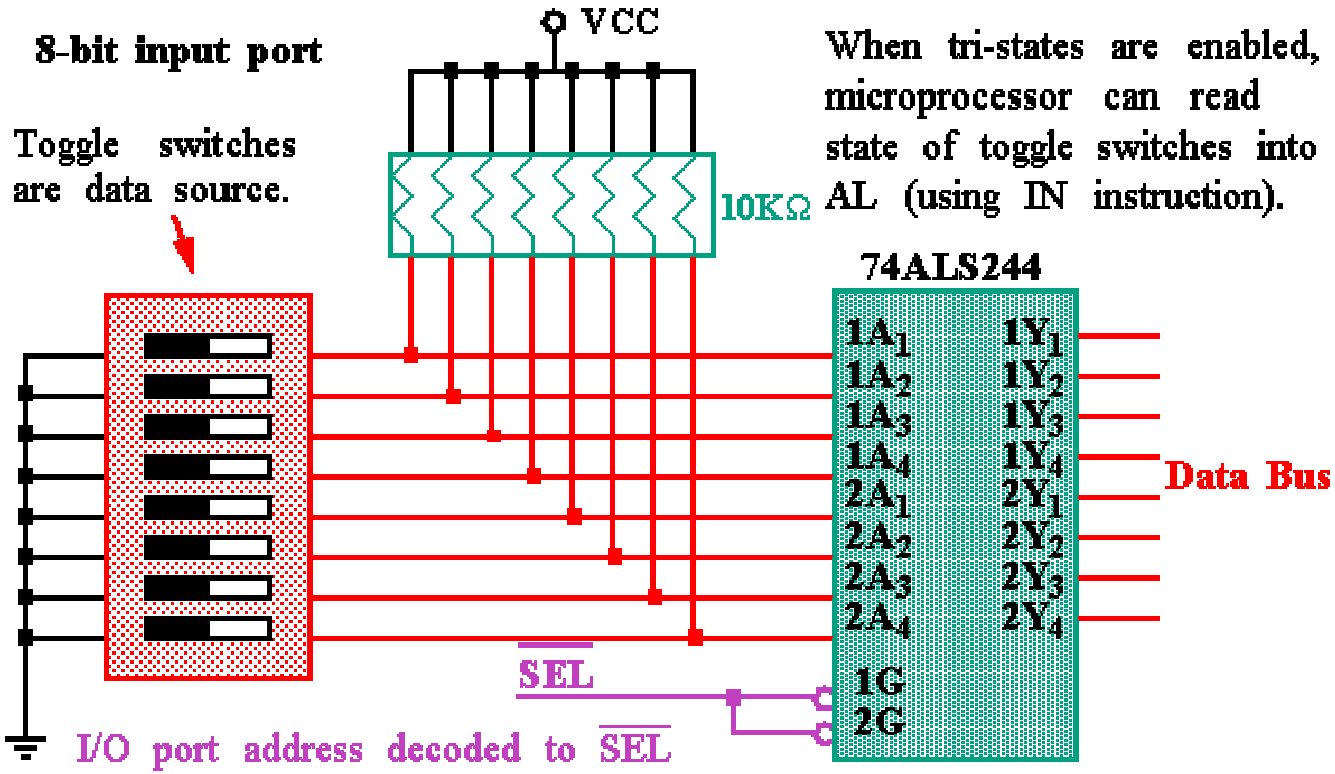
— The decoding circuit controls when the gating device has high impedance output and when it transfers input data to data bus

Input Port Implementation

- Circuit Implementation
  - Assume that the address of the input port is 9CH

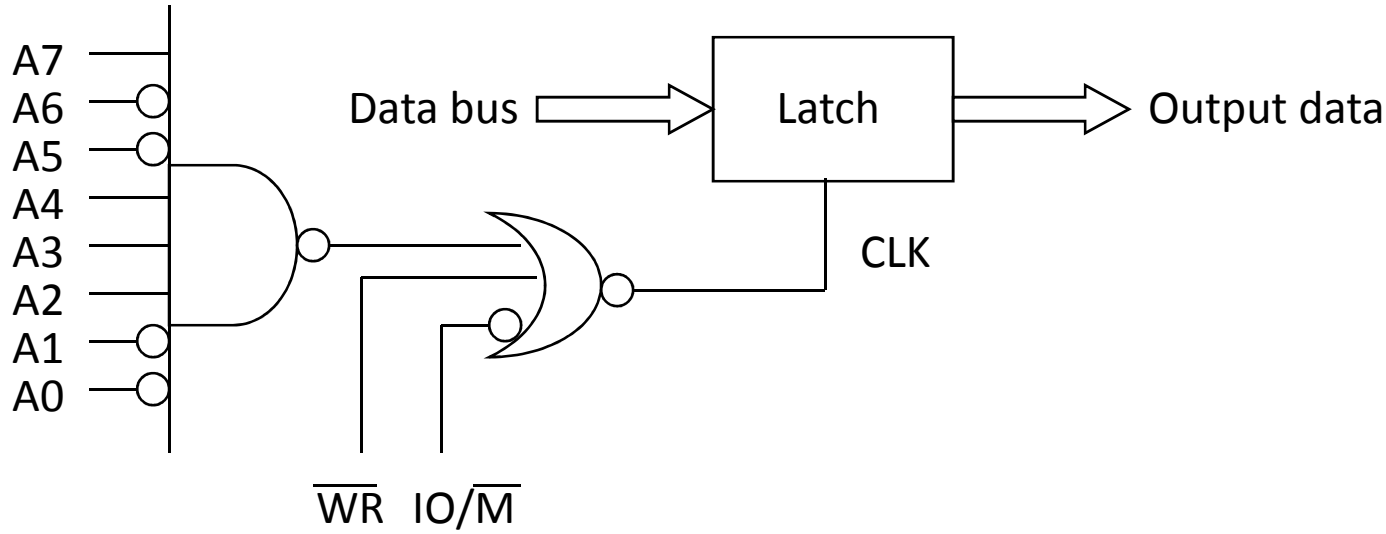


Input Port Implementation

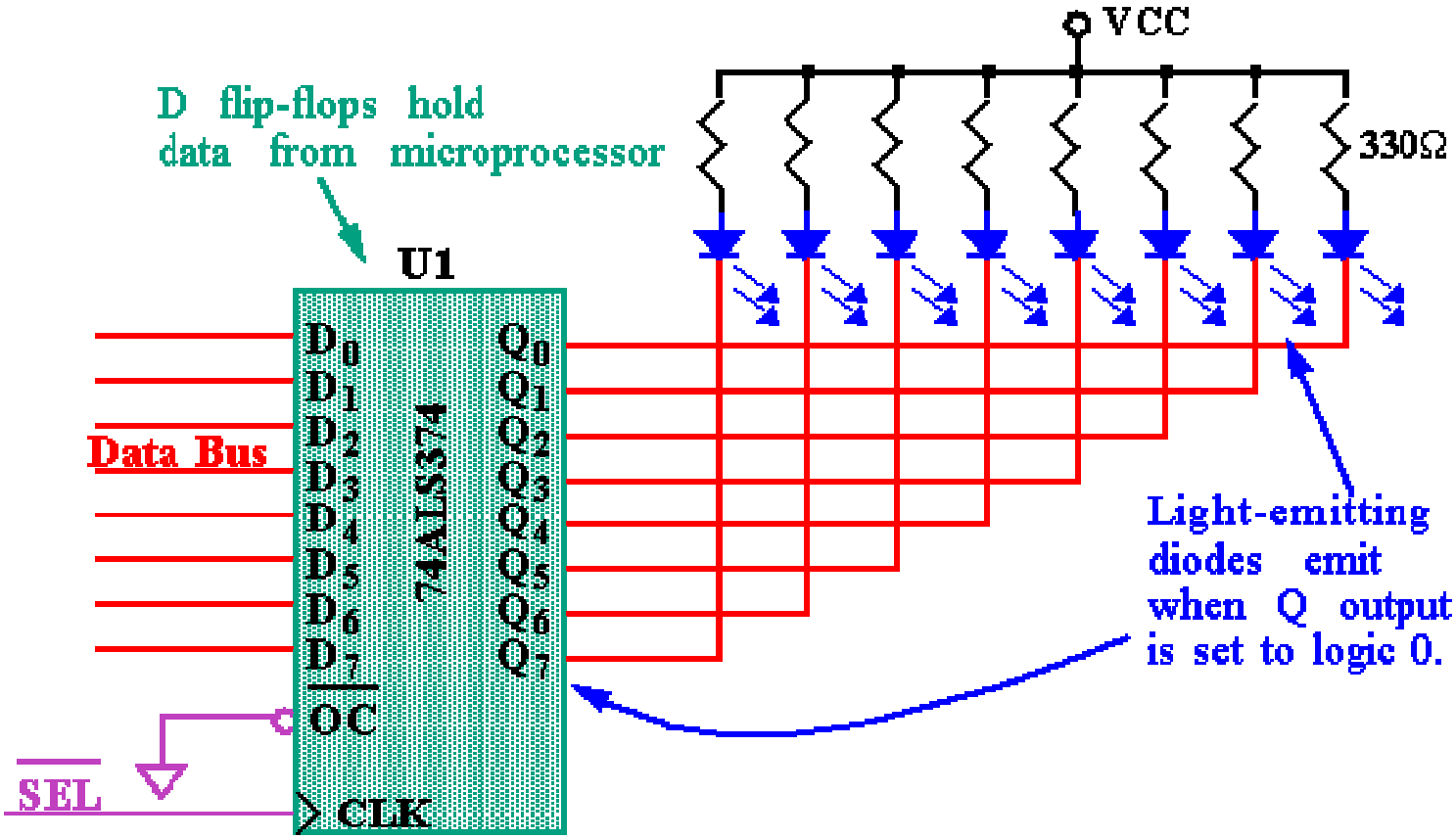


Output Port Implementation

- Circuit Implementation
  - Assume that the address of the output port is 9CH



Output Port Implementation



Decoder generates active low outputs for ports F0H–F7H.

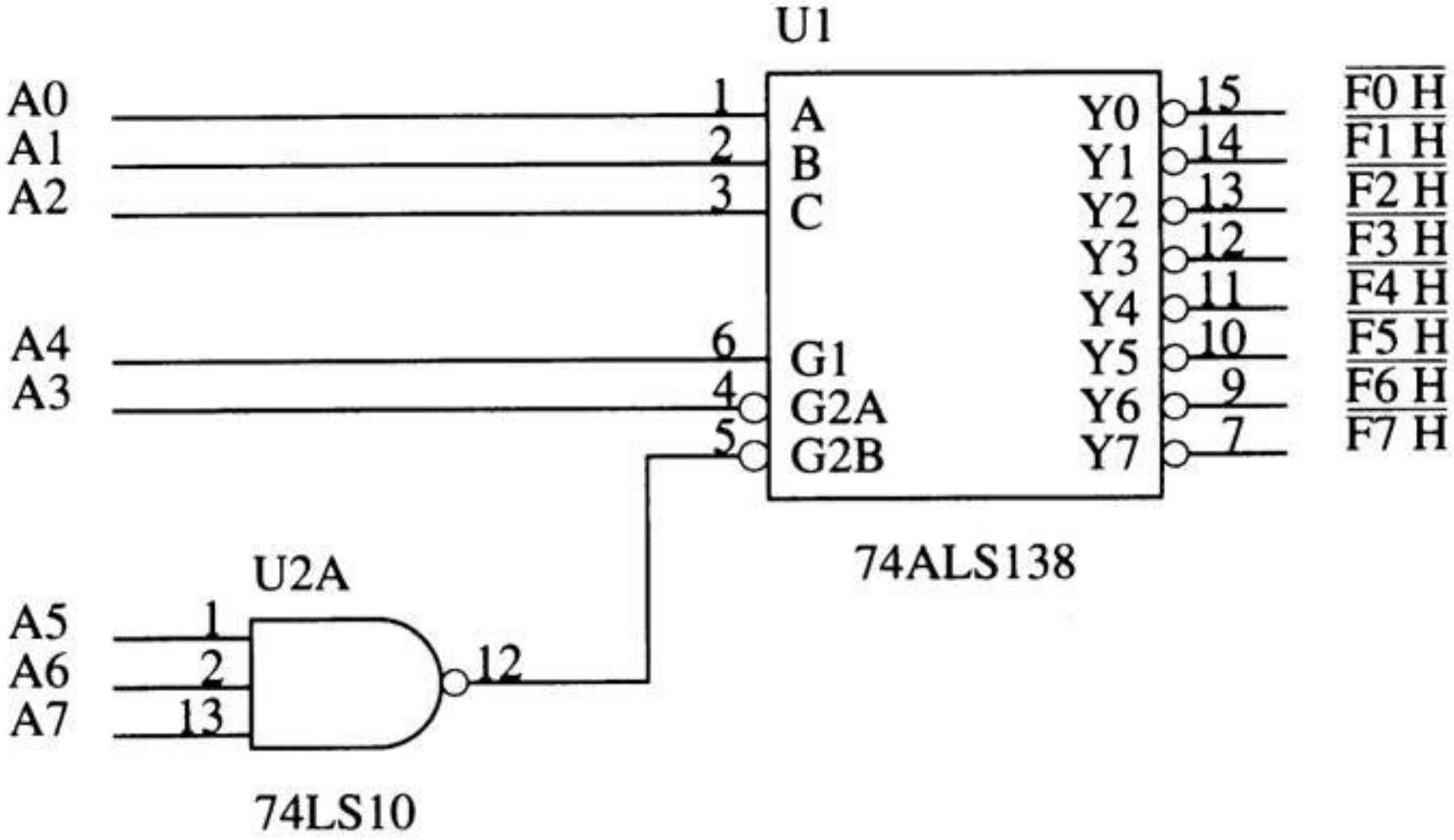
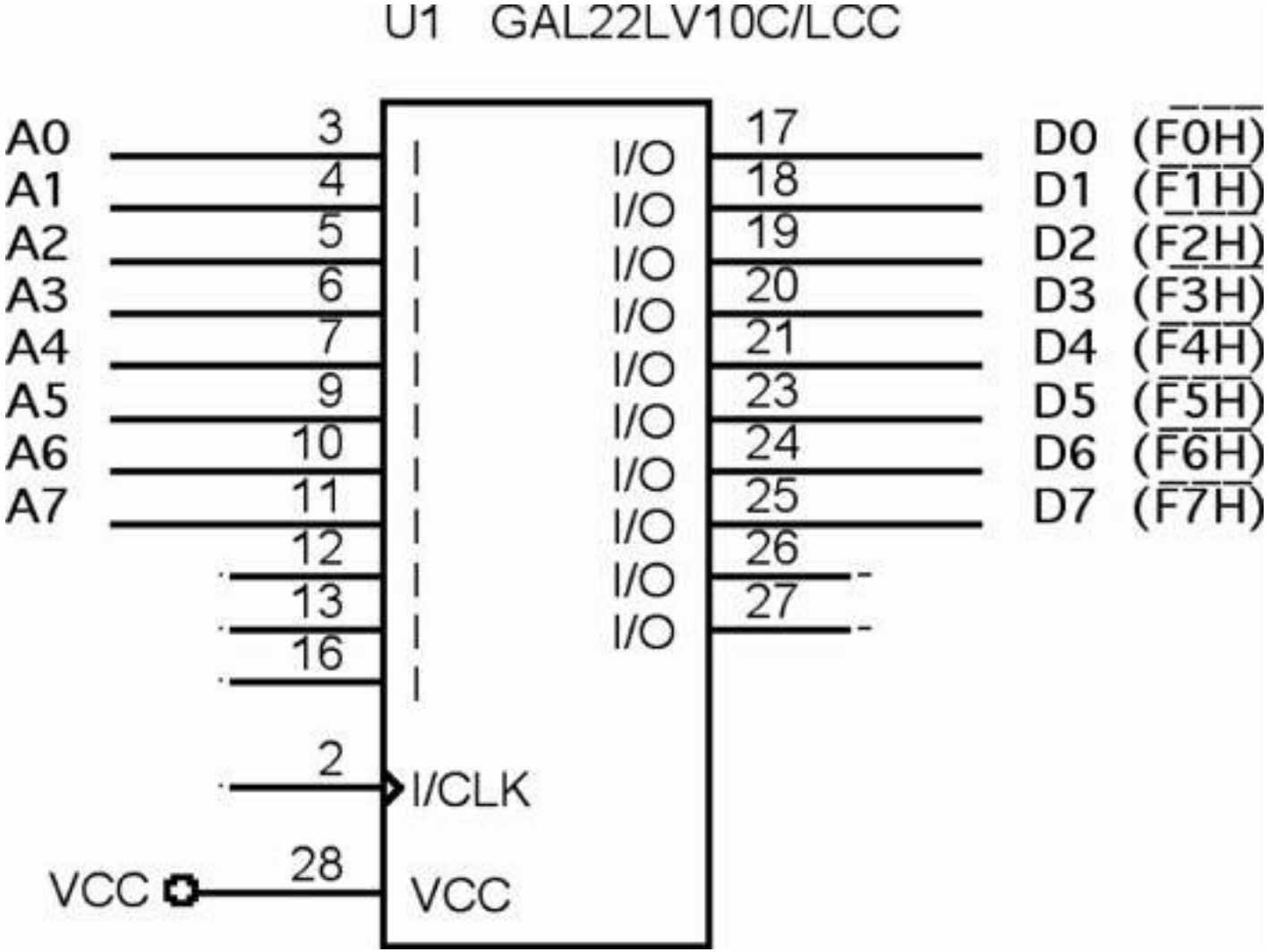


Figure A PLD that generates part selection signals



A PLD that decodes 16-bit I/O ports EFF8H through EFFFH.