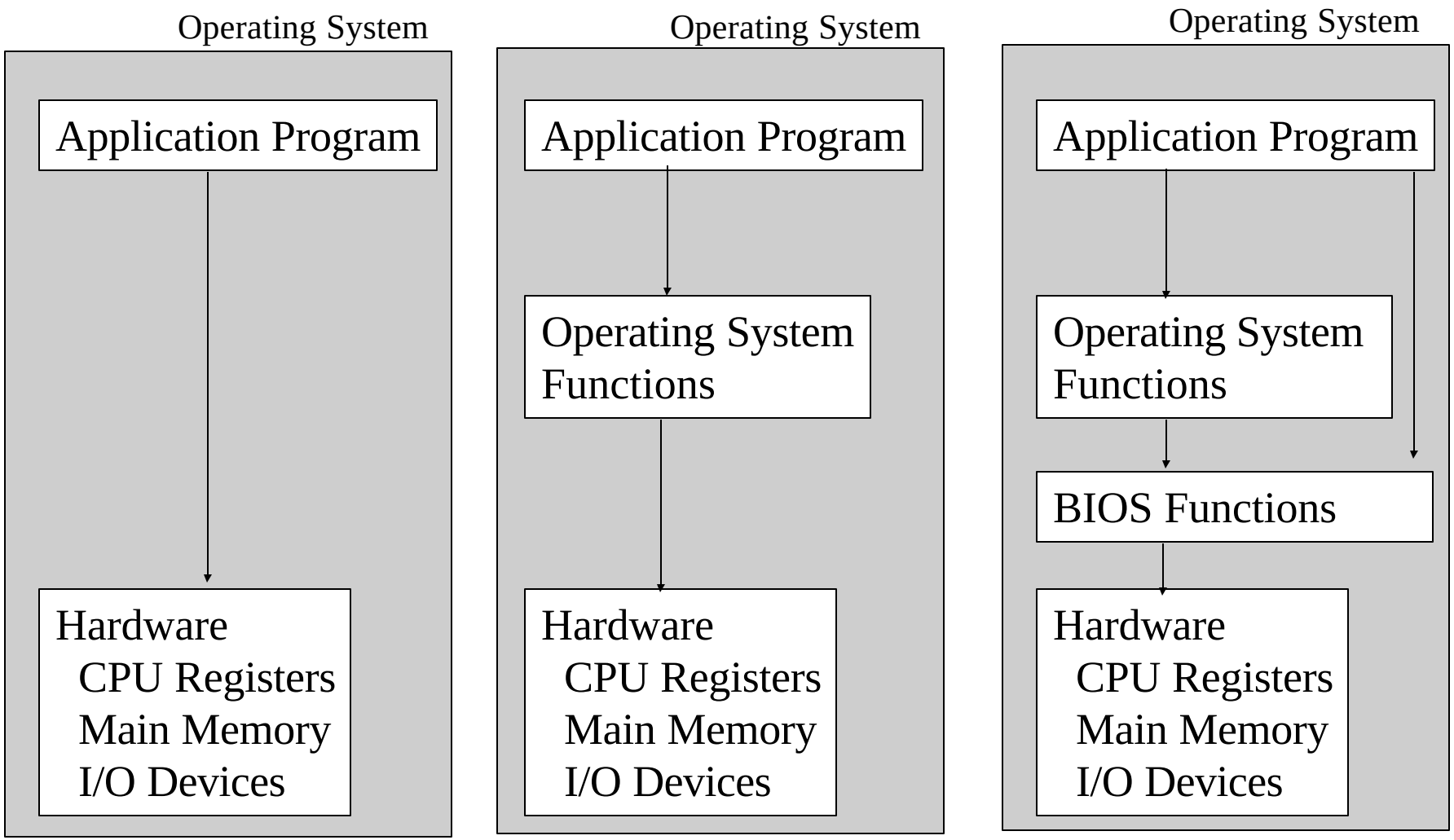
Software Interrupts

Software Interrupts

- **Definition** : A software interrupt is a special call to a procedure previously defined as part of the Operating System
  - Alternate Terminology : TRAP, System Call
  - Implemented using a hardware mechanism: interrupt service routine (ISR).

- **Examples** :
  - DOS Functions
    - Print a string message
    - Exit
    - Character Input
    - Printer Output
  - BIOS Interrupts
    - Video Display Functions
    - Disk I/O functions
    - Keyboard functions
    - Printers functions

Running a Program : Running at Different Levels



Application Subroutines versus Traps

- Application Subroutines:
  - part of a program developed by programmer
  - During assembly: *assembler* determines the target's address offset
    - Static address determined at assembly time
  - Application *linked* with other code (including libraries)
    - These subroutines also been assembled (static addresses)
  - *Loading*: segment values initialized (may be different each time)
    - offset values: static
  - Changes made in the application/libraries, entire application must be re-built :
    - Must be assembled, linked and loaded again. Why ?

Application Subroutines versus Traps

- Traps: Subroutines in the Operating System
  - Transfer control to encapsulated activity terminated by a return to the invocation point.
- Differences w/application subroutines
  - Not part of the application
  - Not part of the program development process
    - OS is not a software library.
    - Program's OBJ file not linked with OS
- Key Difference: OS permanent resident of memory; application is temporary resident

Application program NOT LINKED with the OS. How to "call" the OS Subroutines?

- **Use Vector Table**: array of addresses at "reserved" global location with addresses of the OS procedures.
  - Analogy: Indirect Memory Addressing.
  - OS puts pointers to OS procedures in the array when loaded
  - Applications: developed assuming these global variables exist
  - Activated using existing hardware (built for HW interrupts)
  - What if the OS procedure is revised ?

Software Interrupt (or TRAP) Instruction : INT

- Invoke an ISR: cannot simply use the CALL instruction
  - ISR is identified by interrupt-type (0..255), not a label

- *Software Interrupt* Instruction
  - dynamic subroutine invocation based on stored pointer
  - On the 8086, it is the `INT` instruction.

- Syntax : `INT i`
  - where i = index into the vector table (0..255)
  - executes subroutine with address stored in i-eth position of Vector Table
  - Based on globally assumed vector table at 0:0

Example : Software Interrupt (or TRAP)

- `INT 5`
  - Vector table starts at element 0
  - Executes ISR whose FAR address stored in 6th element of vector table
    - Address of entry = 0:14h (5*4=20)

Subroutines versus Software Interrupts

- Subroutines/Traps similar: both transfer control to encapsulated activity terminated by return to the invocation point.

```
; subroutine initialization
    none
    . . .
; call set up
    push arguments
    CALL subr
    ADD SP, 2*numArgs
    . . .
subr:
    standard entry code
    access param's [BP + 4+]
    standard exit code
    RET
```

n=0..255

```
; ISR initialization
   install address at 0:4*n
   . . .
; call set up
   push arguments
   INT n
   ADD SP, 2*numArgs
   . . .
subr:
   standard entry code
   access param's [BP + 8+]
   standard exit code
   IRET
```

We've already been using DOS Functions

What can you say about parameter passing to DOS Functions ?

```
.data
message db "Hello, world!", 0dh, 0ah, '$'

.code
   ; Print a string
      MOV     AH, 9
      MOV     DX, OFFSET message
      INT     21h
  ; Exit to DOS
      MOV     AX, 4C00h
  ; AH = 4Ch
  ; AL= 0 (exit status)
      INT     21h
```

INT 21h is the DOS Function Call

How does this one ISR handle all the services?

- The type of service is passed in as a parameter
- But parameters are passed by register, not the stack!!

`AH=1`	Keyboard input
`AH=2`	Character output
`AH=5`	Printer output
`AH=9`	String output
`AH=4Ch`	Terminate program

- What about ISR return-value?
  - Depends on the service code parameter!

BIOS Function Calls

Other examples: BIOS Interrupts

- INT 10h for video display functions
- INT 13h for disk I/O functions
- INT 16h for keyboard functions
- INT 17h for printers functions