Computer Architecture summary 1 Cheat Sheet

by pisceswolf96 (pisceswolf96) via cheatography.com/33980/cs/11961/

Programming I/O Devices

What is I/O?

> I/O is the communication between an information processing system, such as a computer, with the outside world, possibly a human or another system. Inputs are signals or data received by the system, and outputs are signals or data sent by it.

What are the components of I/O device?

> There are 2 main components of I/O Device: **Hardware Interface** and **Internal Structure**.

What is hardware structure?

> Hardware Structure is the device interface. it represents the interface used by the system to control the device, and every device has an **interface** and a **protocol** for typical interaction.

How many registers are used in the hardware of an I/O device?

> There are 3 registers in use: **Status Register**, used to get the status of the device, **Command Register**, used to tell the device to perform a certain task, and **Data Register**, used to pass data to the device, or get data from the device.

What is internal structure of an I/O device?

> Internal Structure is the device abstract implementation, it is device specific(meaning it depends on the device itself). Simple devices need one or few hardware chips to implement their functionality, while more complex devices need a simple CPU(also called a micro-controller), a general purpose memory(either DRAM, SRAM or both), and device specific chips to get their jobs done.

Why are I/O slow

> They are slow for 3 reasons:
> **1-** They are the main way of interaction between the user and the computer, so they can be as fast as the user.
> **2-** Their signal is analog and need to be converted to digital.
> **3-** They depend on mechanical movements which is slow compared to digital signals.

What 2 components each I/O device need?

> Each I/O device must have: a **buffer** and a **status register**.

Define a buffer?

> A buffer is a sequence of memory locations, its size depends on the I/O device, it is used to hold data temporarily until it is processed by the CPU.

Define the status register?

> It is a sequence of bits, its size depends on the I/O device, it is used to report the status of the I/O to the CPU?

What is the most important bit in the status register?

> The most important bit is the **ready bit**.

Programmed I/O Technique

What is the protocol of programmed I/O technique?

> The protocol has 4 steps:
> **1-** poll the device to see if it's ready to receive a command(this is done by repeatedly reading the status register).
> **2-** The OS sends data to the data register.{{n}}**3-** The OS writes a command to the command register, this also implicitly tells the device that the data is present in the data register and that it should begin working on the command.
> **4-** The OS waits for the device to finish by again polling it in a loop to see if it has finished(it may then get an error indicating a success or a failure).

When do we call something PIO(programmed I/O)?

> We call it PIO when the CPU is involved in the movement of the data.

What are the advantages of PIO?

> This technique is simple and working.

What are the disadvantages of PIO?

> This technique wastes a great deal of CPU time just waiting for the potentially slow I/O device to complete its activity.

Write the corresponding Assembly code for (cin >> x) using Programmed I/O method.

```
NotReady:
    IN AL , StatusD1 → The status register of Device #1
    AND AL , MaskD1
    JZ NotReady
    IN AL , BufferD1 → The Buffer of Device #1
    MOV X , AL
```

Write the corresponding Assembly code for (cout<<x) using Programmed I/O method.

```
NotReady:
    IN AL , StatusD2
    AND AL , MaskD2
    JZ NotReady
    MOV AL , X
    OUT BufferD2 , AL
```

Interrupt Driven Technique

What is an interrupt?

> An interrupt is a signal emitted to the processor by software or hardware, to indicate that there is an event(which is temporary) that requires immediate attention.

How does the processor responds to an interrupt?

> It responds by **suspending its current activities**, **saving its state**, and **executing a special function called interrupt handler or Interrupt service routine**.

What do we mean by interrupt service routine?

> Interrupt Service Routine(ISR): is a special function that is called when there is an interrupt, this function is called by the processor and not the programmer.

What values do we need to save when an interrupt happens?

> The typical values include: **condition code flags((flag registers)** and contents of any **register** used by both the interrupted program and ISR

Interrupt Driven Technique (cont)

What values does the processor save?

The processor only saves the **Program Counter(PC) which is also called Instruction Pointer(IP)** and the values of **Processor status register**. Other values must be saved by the programmer.

How does the programmer save values of registers used in an ISR?

He does so by pushing them into the **stack segment** in the start of the ISR and then popping them from the stack at the end of the ISR.

What is the difference between an ISR and a standard subroutine(or a function)

The main differences are:
- An ISR can be called anywhere and anytime.
- An ISR is called by the CPU and not the programmer
- The CPU saves the values of registers and the flags, and restore them after the execution of the ISR

Why do we use PUSH and POP commands instead of MOV command when dealing with the stack?

Because both PUSH and POP need only 1 byte, while MOV requires 2 bytes or more.

What is the difference between RET and IRET commands

The RET command returns the value of PC(or IP) **only**, while IRET returns **both** PC(or IP) and the values of the flags.

What do we mean by interrupt nesting?

Pre-emption of low priority interrupt by another high priority interrupt.

How many types of interrupts are there?

There are 2 types: **Hardware** and **Software** interrupts

What are the types of hardware interrupts?

There are 2 types: **Non-Maskable** and **Maskable**.

What do we mean by Non-maskable Interrupts?

It is a high priority interrupt, usually it has its own pin on the processor. In 8086 processor it is called NMI (Non Maskable Interrupt) pin. It is used for power failure routine(the power off button on a desktop or a laptop).

What do we mean by maskable interrupts?

Interrupts that are in levels, and they depend on the INTR(Interrupt Request) pin. When an interrupt is called the **enable bit** is set to 0, which means we can't send any further interrupts until this interrupt is finished.

Give examples on the interrupts preserved for the processor.

- Interrupt 0: preserved for **divide error(division by zero)**
- Interrupt 1: for single step
- Interrupt 2: preserved for **NMI(Non-Maskable Interrupt)**
- Interrupt 3: preserved for **setting a breakpoint**
- Interrupt 4: preserved for **overflow**.

Define Interrupt Vector Table?

Interrupt Vector Table(IVT): is a data structure that contains the address of interrupt service routine, the IVT is located in the first 1024 bytes(1 KB) of memory, it contains 256, 4-byte interrupt vectors.

Interrupt Driven Technique (cont)

What does each ISR takes 4 bytes in the IVT?

it takes 4 bytes because each ISR address is made of:
- code segment register(16 bit or 2 bytes)
- Instruction Pointer(16 bit or 2 bytes).

What is the use of CLI and STI commands?

- CLI clears the **IF(interrupt flag)**, by setting it to zero and thus interrupts are disabled
- STI sets the **IF(interrupt flag)**, by setting it to 1 and thus interrupts are enabled.

What is the 8259A?

It is a PIC (Programmable Interrupt Device), which allows us to have more than 2 levels of interrupts (It allows to have up to 255 levels), it works by combining multiple interrupt inputs into one output which connects to microprocessor.

What are the main pins in 8259A?

- D0-D7: Bidirectional Data connections.
- IR0-IR7: Interrupt request inputs.
- INT(Output): Connects to the microprocessor INTR pin.
- INTA`(Input): connects to the microprocessor INTA` pin.

Q / Write a program in **interrupt driven** to convert cin >> x to Assembly code, presume that the device we are dealing with is a keyboard, and is device0 (D0)

```
ISRD0: ; Interrupt Service Routine for device 0
CLI ; Clear Interrupt Flag (Set it to zero and don't allow interrupts)
PUSH AX ; Put the value of AX into the stack
IN AL, BUFFD0 ; Put the value of the buffer of device 0 into AL
MOV X, AL ; Put the value of AL into X
POP AX ; Take out the value of AX from the stack
STI ; Set Interrupt Flag (Set it to one and allow interrupts)
IRET ; Interrupt Return (This pops the value of IP, CS and Flags from the stack)
```