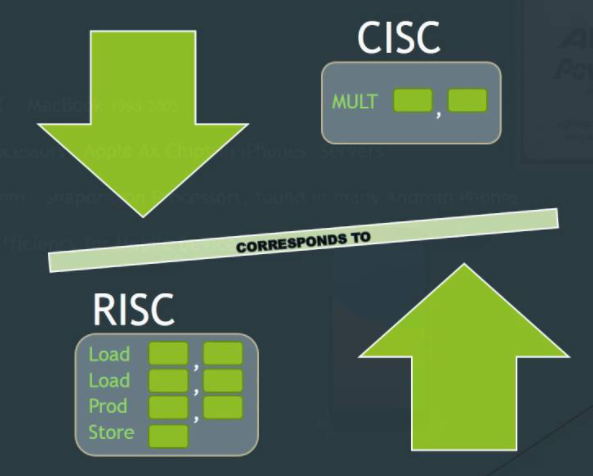
**Lecture 1**

CISC**:**  Complete Instruction set

RISC**:**  Reduced Instruction set

* All instruction in Risc take only 1 cycle
* Micro Controller is processor do only one thing
* Power consumption proportionate to Clock speed

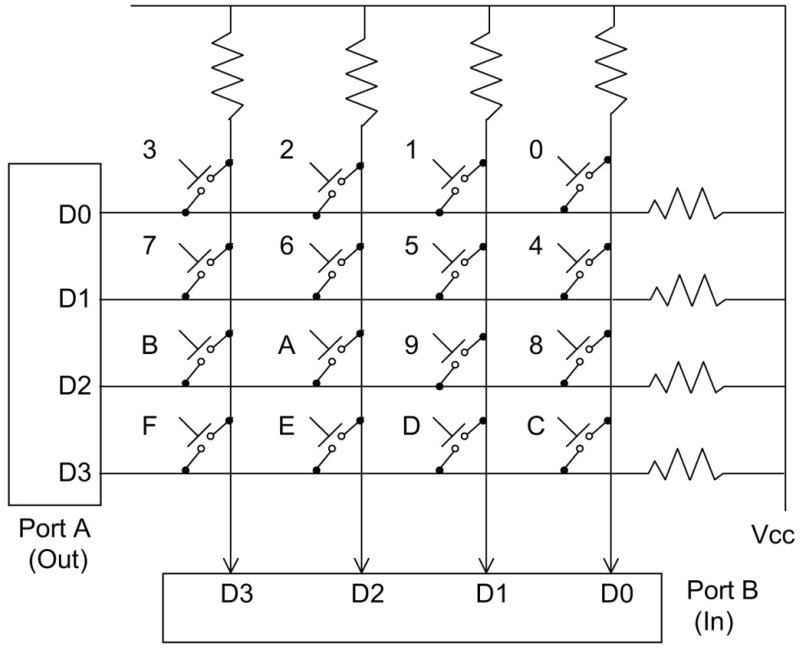
|  |  |
| --- | --- |
| CISC | RISC |
| FEW INSTRUCTIONS | MULTIPLE INSTRUCTIONS |
| LESS REGISTERS | MORE REGISTERS |
| MORE MICROPROGRAMMING | MORE COMPLEX COMPILERS |
| N CYCLE TIMES PER INSTRUCTION | ONE CYCLE TIME PER INSTRUCTION |
| HARDWARE FOCUSED | SOFTWARE FOCUSED |

\*Risc have more registers because we want to make it faster without increase clock Cycle

**KEYBOARD**

-Keyboards are organized in a matrix of rows and columns, and the CPU accesses rows & columns through ports.

-With two 8-bit ports, an 8 x 8 = A \* A matrix of keys can be connected to a microprocessor. A = Log2(n of keys)

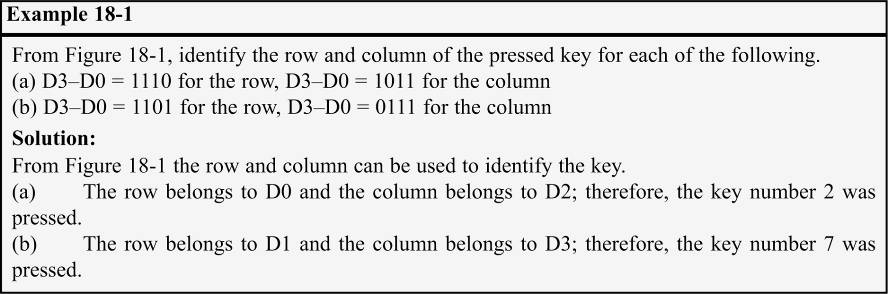
- If no key has been pressed, the input port will yield 1s for all columns, since they are all connected to high. (Vcc)

- at start all Port A is 0

- when key pressed we start (**Detection phase**) port B corresponding to clicked key become 0 due to short circuit Between it and Port A Corresponding

- as ex if we click 9 so B(D1) will become zero but that mot enough because if 1 ,5,D clicked same will happed

- here start (**Identification phase**) we try put 0 to each one Of Port A and see Port B so if Corresponding Port B is 1 so that mean its not right & if is 0 so that mean we are right and this is corresponding key.

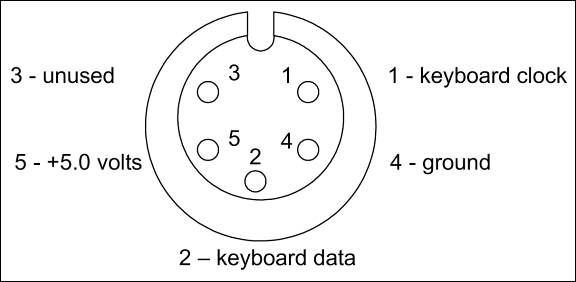
- so to do that we create Micro Controller to do that operation and send it to processor serial to save money because human is so slow compared to computer.

-EX

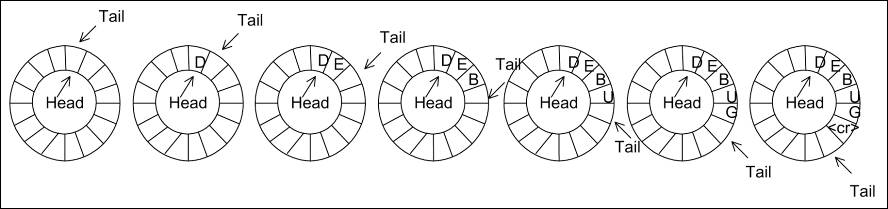
* IBM PC/AT keyboards use the following data frame to send scan code serially to the motherboard.
  + For each scan code, a total of 11 bits are transferred.
  + One start bit (always 0)
  + 8 bits for scan code
    - 7 bit for data
    - 1 bit to detect if it pressed(clicked = 0) or Release(Break = 1)
  + Odd parity bit
  + One stop bit (always 1)
* So each button is sent twice once with clicked & once for Break
* IRQ : interrupt request
  + It’s the priority according to it processor doe which tasks first or right now
  + Higher is zero 0 --> higher

-So all processor will be like

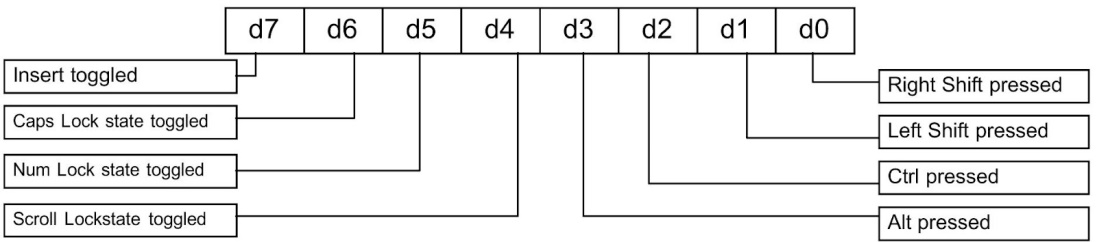
* The keyboard microcontroller scans the keyboard matrix continuously.
* When a key is pressed (a make)…
  + It is identified and its scan code is sent serially (Scan & ASCII Code) to the motherboard through the keyboard cable.
* The circuitry on the motherboard…
  + Receives the serial bits.
  + Gets rid of the frame bits
  + Makes one byte (scan code) with its serial-in-parallel-out shift register
  + Presents this 8-bit scan code to port A of 8255 at I/O addresses 60H.
  + Activates IRQ1.
* Since IRQ1 is set to INT 09, its interrupt service routine (ISR) residing in BIOS ROM is invoked.
  + ISR of INT 09 reads the scan code from port 60H.
  + ISR of INT 09 tests the scan code to see if it is the Right or Left Shift, Alt, Ctrl keys, etc.
    - If so, the appropriate bit of the keyboard status bytes in BIOS 0040:0017H and 0018H are set.
      * It will not write the scan code to the keyboard buffer.
* Before returning from INT 09, ISR will issue EOI(استني) to unmask IRQ1, followed by the IRET instruction.
  + This allows IRQ1 activation to be responded to again
* When the key is released (a break), the keyboard generates the second scan code by adding 80H to it and sends it to the motherboard.
* ISR of INT 09 checks the scan code to see if there is 80H difference between this code and the old one.
  + If D7 is high, it is interpreted as meaning the key has been released &the system ignores the 2nd scan code.
  + If the key is held down more than 0.5 seconds, it is interpreted as a new key and INT 09 will write it into  
    the keyboard buffer (32 bit which mean 16 char only in buffer) next to the preceding one. Commonly referred to as typematic in IBM literature, which means repeating the same key.



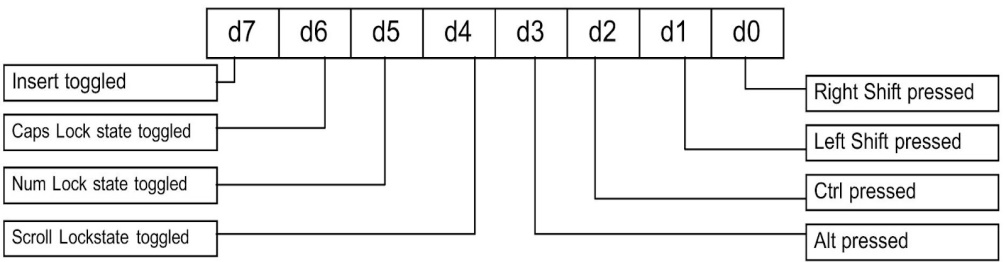
* As INT 16H reads a character from the keyboard buffer, it advances the head pointer, which is held by memory locations 41AH and 41BH.
  + As INT 09 inserts the character into the keyboard buffer, it advances the tail.
  + As INT 16H reads the character from the keyboard buffer it advances the head



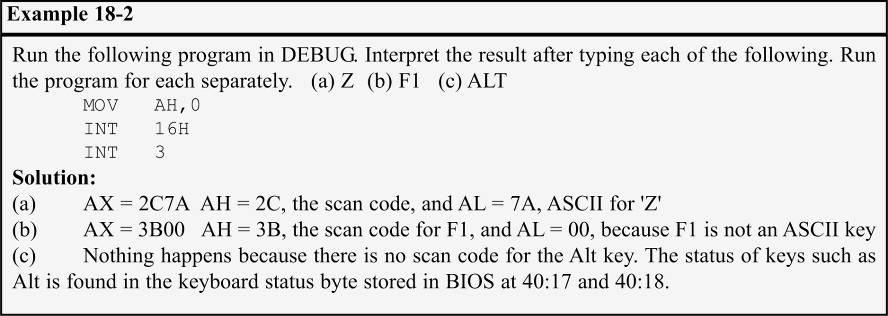
* **keyboard buffer**
* 32 bytes (16 words) of BIOS data memory is set aside, at addresses 40:001EH - 40:003DH.
* Each two consecutive locations are used for a single character.
  + One for the scan code, the other for the ASCII code (if any) of the character.
* When they come to the end of the keyboard buffer, they both wrap around, creating a ring of 16 words where the head is continuously chasing the tail.
* If the buffer is empty, head address equals tail address.
* **Status 2 Byte**



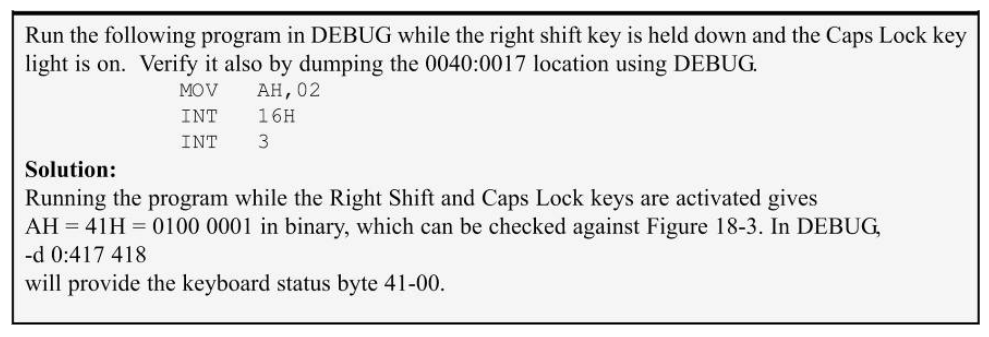
- Location 0040:0017H holds the shift status byte

- Location 0040:0018H holds the second status byte

* INT 16H reads a character from the keyboard
* INT 16H, AH = 0 (read a character)
  + checks the keyboard buffer for a character.
  + If a character is available, it returns its scan code in AH and its ASCII code in AL.
  + If no character is available in the buffer, it waits for a key press and returns it.
  + For characters with no ASCII code, it provides the scan code in AH and AL = 0. Such as F1–F10.



* INT 16H, AH = 01 (find if a character is available) - checks the keyboard buffer for a character.
  + If a character is available, it returns its scan code in AH, its ASCII code in AL, and sets ZF = 0.
  + If no character is available in the buffer, it does not wait for a key press, and simply makes ZF = 1.
* INT 16H, AH = 02 (return current keyboard status byte) - provides keyboard status in register AL.
  + The keyboard status byte (also referred to as the keyboard flag byte) is located in the BIOS data area memory location 0040:0017H.

****