** NATIONAL UNIVERSITY OF COMPUTER & EMERGING SCIENCES**

**PROJECT:**  
 COMPUTER ORGANIZATION & ASSEMBLY LANGUAGE **(COAL)**

**TOPIC:**

DIGITAL CLOCK

**GROUP MEMBERS:**

* + - ASJID TAHIR **(19p-0085)**
    - IBRAR BABAR **(19p-0104)**
    - USMAN MANZOOR **(19P-0068)**

**Digital Clock:**

A **clock** or **watch** in which the **hours, minutes**, and sometimes **seconds** are indicated by digits. It shows the time using numbers, not hands. Digital clocks are often associated with electronic drives, but the “**digital**” description refers only to the display, not the drive mechanism. To represent the time, most digital clocks use a seven-segment display. It shows hours, minutes and seconds and can be set to show in both **24 hours** and **12 hours** format.

**Features of Digital Clock:**

Following are the basic features of digital clock:

* It shows **hours**, **minutes** and **seconds.**
* It uses **12 hour** format.
* It is developed in VGA graphics with 320x200 resolution.
* It is displayed in a **classy**, **stylish** and **attractive** font.
* The interface is simplistic with white text on black background.

**Functions:**

* **Start:**  
  Extra segment (**es**), which provide additional segments for storing data.
* We cannot move value directly to **ex** register.
* So we first store value to **ax** and then **mov ax to es**.
* **ES** is the value where we start reading.
* **Space key for Starting:**
* Press space for starting of digital clock use for starting the digital clock **mov ax,[es:9\*2]** which is actually used to move the value from which reading is going to start.
* And **9\*2** is for **index**, which further on is used for accessing the next indexes.
* And at the last **[es:9\*2]** is move in **ax** register.

**oldkb**, which is declared as a global variable having data type, **(dd**), is to store **ax** value**. mov ax, [es:9\*4+2],** which is going to store next index value in **ax** register, and then store it in the **oldkb** register.

* **Call ClearScreen (hading):**
* In clear screen function we have to **move 0xb800**, which is a specific location in memory in which the higher **4000** bytes is used for some specific task.
* In other words, these **4000** bytes is actually contain the dosbox size**. mov es, ax,** **xor di, di**
* mov **ax**, 0x720
* mov **cx**, 2000.
* In these instructions, we simply **mov ax in es**, as we cannot move value directly to **es** register. and next we **mov ax,0x0720**, in which (07) is used for black background and (20) is used for space. and then simply **move 2000 to cx**, which is used as **counter**.
* **cld**, this instruction is used for auto-increament in **cx** times. rep stosw, which is the combination of source and destination (**source=ax**)
* [**destination=es**].
* then pop instructions and **ret**.
* **CheckLMode1:**
* **cmp al, 182** (for storing the next loop again) **jnz startTime**, if **jmp** is not equal to zero then go to startTime and do the following operations.
* else **mov byte [cs:sMode], 0** (Disable the split Mode, it is same as the upper label, in which **cs** is used as index
* And only 1 byte is given to it, in other words we can say that we **mov 0** to that location of 1 byte).
* **cmp byte [cs:lMode], 1** (If **lMode** was already enabled then do nothing, it mean you don’t have to do anything, that's flag).
* **jmp EOI1** (e0i1 is explained below).
* **Space k release:**
* startTimeL: **cmp byte [cs:startTimer], 1** (If Timer is already started then do nothing)
* **jz check0** (check zero will be explained below**)**
* **mov byte [cs:startTimer], 1** (Else start the digital clock start).
* **Check0:**
* **as** check0 label is explained in the above labels. is actually go to silt the digital clock time like if 60 seconds go for completion then other increament in minute portion and start again seconds modes again.
* **cmp byte [cs:sMode], 1**.If split mode is enabled.
* **Check1:**
* **as** check1 label is explained in the above labels. It is actually go to enable the lap mode when each itration is executed.
* **cmp byte** **[cs:lMode], 1**.
* Else if the lap mode is enabled.
* **EOI1:**
* **as** EOI1 label is mentioned in the above labels, in which we use the instruction which is **ret**, which reverses the operation of an **INT** or **CALL** that caused the task switch if **NT** equals.
* **Print time on dosbox:**
* When print function is **call** from main then first we have to push bp on the stack and then we make **bp** and **sp** equal in order to access other values on the **stack**.
* Then **es** register push and **move 0xb800** in **ax** which is the starting address of dosbox, then **move ax to es**, as we cannot move address or value directly to **es** register.
* (**di**) is used as index (**mov di, [bp+4]**);Location where the time is to be printed.
* **Hours time** is to be printed at location **[bp+6],** and **call** printstr function which is already discuss above.
* After hours we have to print **colon**, for this we use instruction, **mov byte [es:di], ':**'.
* **Minutes time** is to be printed at location **[bp+8],** and **call** printstr function which is already discuss above.
* After **minutes**, we have to print **colon**, for this we use instruction, **mov byte [es:di], ':'**.
* **Seconds time** is to be printed at location **[bp+10],** and **call** printstr function which is already discuss above.
* then simple pop **es** and pop **bp**.
* **Next Digit:**
* In the next digit, we move **zero** to **dx** register.
* Then divide the register bx and add the **dl** to set the time position and then we push the register **dx** into the **stack** and we increament the value of cs and now we can compare it **if jump is not equal to zero** move to the next digit otherwise move to the next position function which is explained below and then compare it if jump is not equal to the 1 and then we **mov byte [es:di], '0'** di used for **indexing** and **es** is register and we can give the jump of **2** and print it.
* **Next Position:**
* In the nextpos, we first pop the **dx** register.
* And then we move **dh to 0x07** due to we use black screen in **dosbox** to print our output which is time in **white display** thats we move this address to this.
* And then we **mov [es:di], dx** dx register into the location address **di** where it is point in the **es** and give the **jump of 2** and print it and continue our loop. and then we pop **es** and a and then **bp** and then we **return 4** to pop all the elements from the stack.