Department of Engineering Technology

Foundation University Islamabad

School of Science and Technology

Computer Architecture and Assembly Language Lab

Names: Haris Hayat- FA23-BSIET-078

Sohaib Umar Ranjha FA23-BSIET-084

Saad Ahmed FA23-BSIET-090

Shaban Saeed FA23-BSIET-082

PROGECT REPORT

OBJECTIVES:

- **Explained working of code in steps.**
- **Output Explanation.**
- **Short Summary of code.**

Remarks (if any): …….

Name & Signature of faculty: ……

REPORT

Step 1: Initialize Data Segment and Stack

1. .MODEL SMALL specifies the memory model used by the program.
2. .STACK 100H allocates 100H bytes of stack space.
3. .DATA begins the data segment, where variables are defined.
4. PROMPT DB 'Current System Time is : $' defines a string variable PROMPT with the value"Current System Time is : ".
5. TIME DB '00:00:00$' defines a string variable TIME with the initial value "00:00:00".

```
.MODEL SMALL
.STACK 100H
.DATA
PROMPT  DB 'Current System Time is : $'
TIME DB '00:00:00$'
```

Step 2: Initialize Code Segment

1. .CODE begins the code segment, where the program's instructions are defined.
2. MAIN PROC defines the main procedure of the program.

```
.CODE
  MAIN PROC
```

Step 3: Initialize Data Segment Register

1. MOV AX, @DATA loads the address of the data segment into the AX register.
2. MOV DS, AX sets the DS (Data Segment) register to the value in AX, which points to the data segment.

```
   MOV AX, @DATA
   MOV DS, AX
```

Step 4: Load Address of TIME Variable

1. LEA BX, TIME loads the address of the TIME variable into the BX register.

```
   LEA BX, TIME
```

Step 5: Call GET_TIME Procedure

1. CALL GET_TIME calls the GET_TIME procedure to retrieve the current system time.

```
CALL GET_TIME
```

Step 6: Display Prompt Message

1. LEA DX, PROMPT loads the address of the PROMPT message into the DX register.
2. MOV AH, 09H sets the AH register to 09H, which is the function code for displaying a string.
3. INT 21H calls the DOS interrupt 21H, which displays the string pointed to by DX.

```
LEA DX, PROMPT
MOV AH, 09H
INT 21H
LEA DX, TIME
MOV AH, 09H
INT 21H
```

Step 7: Display Current System Time

1. LEA DX, TIME loads the address of the TIME variable into the DX register.
2. MOV AH, 09H sets the AH register to 09H, which is the function code for displaying a string.
3. INT 21H calls the DOS interrupt 21H, which displays the string pointed to by DX.

```
LEA DX, TIME
MOV AH, 09H
INT 21H
```

Step 8: Terminate Program

1. MOV AH, 4CH sets the AH register to 4CH, which is the function code for terminating a program.
2. INT 21H calls the DOS interrupt 21H, which terminates the program.

GET_TIME Procedure

3. PUSH AX and PUSH CX save the AX and CX registers on the stack.
4. MOV AH, 2CH sets the AH register to 2CH, which is the function code for getting the current system time.
5. INT 21H calls the DOS interrupt 21H, which returns the current system time in the CH, CL, and DH registers.

```
MOV AH, 4CH
INT 21H
MAIN ENDP
GET_TIME PROC
PUSH AX
PUSH CX
MOV AH, 2CH
INT 21H
```

Step 9: Convert Hours to ASCII

1. MOV AL, CH moves the hours value from CH to AL.
2. CALL CONVERT calls the CONVERT procedure to convert the hours value to ASCII.
3. MOV [BX], AX stores the converted hours value in the TIME buffer at offset 0.

```
MOV AL, CH
CALL CONVERT
MOV [BX], AX
```

Step 10: Convert Minutes to ASCII

1. MOV AL, CL moves the minutes value from CL to AL.
2. CALL CONVERT calls the CONVERT procedure to convert the minutes value to ASCII.
3. MOV [BX + 3], AX stores the converted minutes value in the TIME buffer at offset 3.

```
MOV AL, CL
CALL CONVERT
MOV [BX + 3], AX
```

Step 11: Convert Seconds to ASCII

1. MOV AL, DH moves the seconds value from DH to AL.
2. CALL CONVERT calls the CONVERT procedure to convert the seconds value to ASCII.
3. MOV [BX + 6], AX stores the converted seconds value in the TIME buffer at offset 6.

```
MOV AL, DH
CALL CONVERT
MOV [BX + 6], AX
```

Step 12: Restore Registers and Return

1. POP CX and POP AX restore the CX and AX registers from the stack.
2. RET returns from the GET_TIME procedure.

CONVERT Procedure

3. PUSH DX saves the DX register on the stack.
4. MOV AH, 0 sets the AH register to 0.
5. MOV DL, 10 sets the DL register to 10.
6. DIV DL divides the value in AX by 10, separating the tens and units digits.
7. OR AX, 3030H converts each digit to ASCII by adding 30H (the ASCII code for '0') to the result.
8. POP DX restores the DX register from the stack.
9. RET returns from the CONVERT procedure.

```
POP CX
POP AX
```

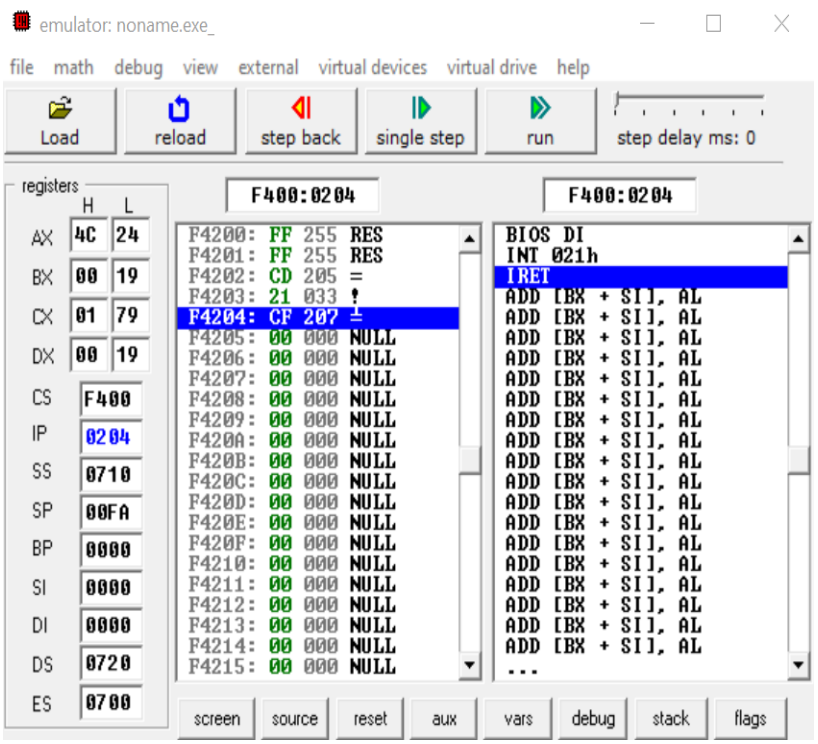
```
    RET
GET_TIME ENDP
   CONVERT PROC
    PUSH DX
    MOV AH, 0
    MOV DL, 10
    DIV DL
    OR AX, 3030H
    POP DX
    RET
   CONVERT ENDP
 END MAIN
```

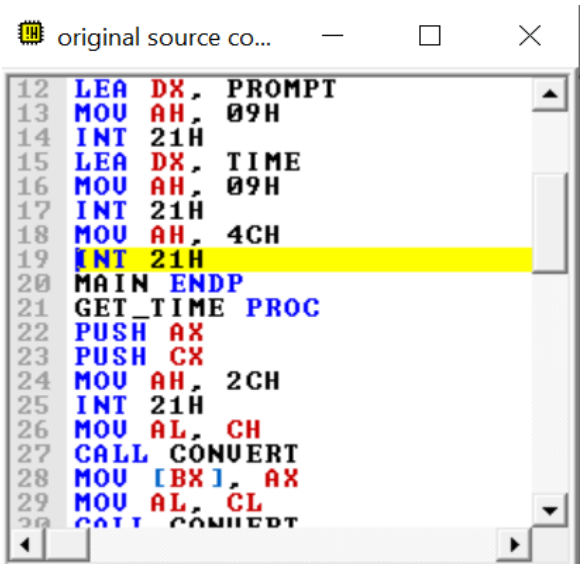
CODE

```
.MODEL SMALL
.STACK 100H
.DATA
PROMPT  DB 'Current System Time is : $'
TIME DB '00:00:00$'
.CODE
   MAIN PROC
     MOV AX, @DATA
     MOV DS, AX
     LEA BX, TIME
     CALL GET_TIME
     LEA DX, PROMPT
     MOV AH, 09H
     INT 21H
     LEA DX, TIME
     MOV AH, 09H
     INT 21H
     MOV AH, 4CH
     INT 21H
   MAIN ENDP
GET_TIME PROC
    PUSH AX
    PUSH CX
    MOV AH, 2CH
    INT 21H
    MOV AL, CH
    CALL CONVERT
    MOV [BX], AX
    MOV AL, CL
    CALL CONVERT
    MOV [BX + 3], AX
    MOV AL, DH
```

```
    CALL CONVERT
    MOV [BX + 6], AX
    POP CX
    POP AX
    RET
GET_TIME ENDP
  CONVERT PROC
    PUSH DX
    MOV AH, 0
    MOV DL, 10
    DIV DL
    OR AX, 3030H
    POP DX
    RET
  CONVERT ENDP
 END MAIN
```

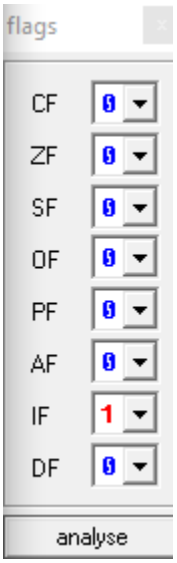
OUTPUT :






emulator screen (80x25 chars)
Current System Time is: 20:54:10

FLAGS:


flags
CF 0
ZF 0
SF 0
OF 0
PF 0
AF 0
IF 1
DF 0
analyse