CSPC22 - Lab Assignments

Name: Priyanshu Tripathi Roll Number: 12022005 Section: CS-B5

Submitted to: Mohit Dua & Nidhi Chakravarty Submitted to: 27/04/2022

Index

Contents

[Experiment 1 - Printing 3](#_Toc101952726)

[A-Z by character 3](#_Toc101952727)

[A-Z by ASCII 3](#_Toc101952728)

[a-z by character 3](#_Toc101952729)

[a-z by ASCII 3](#_Toc101952730)

[Display 0 - 9 4](#_Toc101952731)

[Display the ASCII table 4](#_Toc101952732)

[Display AaBbCc… 4](#_Toc101952733)

[Display AaaBbbCcc… 4](#_Toc101952734)

[Display AbCdEf… 5](#_Toc101952735)

[Experiment 2 – Strings 6](#_Toc101952736)

[Print using 09h 6](#_Toc101952737)

[Print string character-wise 6](#_Toc101952738)

[Program a. in 16-bit 6](#_Toc101952739)

[Program b. in 16-bit 7](#_Toc101952740)

[Print a string in reverse 7](#_Toc101952741)

[Print a string in reverse in 16-bit 8](#_Toc101952742)

[Check if a string is a palindrome or not 8](#_Toc101952743)

[Experiment 3 – Arithmetic 10](#_Toc101952744)

[Add two 8-bit single digit numbers 10](#_Toc101952745)

[Add two 16-bit single digit numbers 10](#_Toc101952746)

[Subtract two 8-bit single digit numbers 11](#_Toc101952747)

[Subtract two 8-bit multi digit numbers 12](#_Toc101952748)

[Multiply two 8-bit single digit numbers 14](#_Toc101952749)

[Multiply two 16-bit single digit numbers 14](#_Toc101952750)

[Divide two 8-bit single digit numbers 15](#_Toc101952751)

[Experiment 5 – Integer Properties 17](#_Toc101952752)

[Check if an 8-bit number is positive or negative 17](#_Toc101952753)

[Check if a 16-bit number is positive or negative 17](#_Toc101952754)

[Check if an 8-bit number is odd or even 18](#_Toc101952755)

[Check if a 16-bit number is odd or even 19](#_Toc101952756)

[Experiment 7 – Factorial 20](#_Toc101952757)

[Experiment 8 – Fibonacci 21](#_Toc101952758)

# Experiment 1 - Printing

## A-Z by character

|  |  |
| --- | --- |
| start: mov cl,26 mov dl,’A’ L:  mov ah,02h  int 21h  inc dl  Loop L end start | Graphical user interface, application  Description automatically generated |

## A-Z by ASCII

|  |  |
| --- | --- |
| start: mov cl,26 mov dl,65 L:  mov ah,02h  int 21h  inc dl  Loop L end start | Graphical user interface, application  Description automatically generated |

## a-z by character

|  |  |
| --- | --- |
| start: mov cl,26 mov dl,’a’ L:  mov ah,02h  int 21h  inc dl  Loop L end start | Graphical user interface, application  Description automatically generated |

## a-z by ASCII

|  |  |
| --- | --- |
| start: mov cl,26 mov dl,97 L:  mov ah,02h  int 21h  inc dl  Loop L end start | Graphical user interface, application  Description automatically generated |

## Display 0 - 9

|  |  |
| --- | --- |
| start: mov cl,10 mov dl,'0' L:  mov ah,02h  int 21h  inc dl  loop L  end start | Graphical user interface, application  Description automatically generated |

## Display the ASCII table

|  |  |
| --- | --- |
| start: mov cx,256 mov dl,0 L:  mov ah,02h  int 21h  inc dl  loop L  end start | Graphical user interface, application  Description automatically generated |

## Display AaBbCc…

|  |  |
| --- | --- |
| .code  start:      mov dl, 'A'      mov cl, 26      L:          mov ah, 02h          int 21h          add dl, 32          mov ah, 02h          int 21h          sub dl, 32          inc dl          loop L  end start |  |

## Display AaaBbbCcc…

|  |  |
| --- | --- |
| .code  start:      mov dl, 'A'      mov cl, 26      L:          mov ah, 02h          int 21h          add dl, 32          mov ah, 02h          int 21h          mov ah, 02h          int 21h          sub dl, 32          inc dl          loop L  end start | Graphical user interface, application  Description automatically generated |

## Display AbCdEf…

|  |  |
| --- | --- |
| .code  start:  mov dl, 'A'  mov cl, 13  L:  mov ah, 02h  int 21h  inc dl  add dl, 32  mov ah, 02h  int 21h  sub dl, 32  inc dl  loop L  end start | Graphical user interface, application  Description automatically generated |

# Experiment 2 – Strings

## Print using 09h

|  |  |
| --- | --- |
| .data  string db "Hello, World!$"  .code  start:  mov ax, @data  mov ds, ax  mov dl, offset string  mov ah, 09h  int 21h  end start | Graphical user interface, application  Description automatically generated |

## Print string character-wise

|  |  |
| --- | --- |
| .data  string db "Hello, World!$"  .code  start:  mov ax, @data  mov ds, ax  mov si, offset string  L:  mov dl, [si]  mov ah, 02h  int 21h  inc si  cmp [si], '$'  jnz L  end start | Graphical user interface, application  Description automatically generated |

## Program a. in 16-bit

|  |  |
| --- | --- |
| .data  string dw "Hello, World!$"  .code  start:  mov ax, @data  mov ds, ax  mov dx, offset string  mov ah, 09h  int 21h  end start | Graphical user interface, application  Description automatically generated |

## Program b. in 16-bit

|  |  |
| --- | --- |
| .data  string dw "Hello, World!$"  .code  start:  mov ax, @data  mov ds, ax  mov si, offset string  L:  mov dx, [si]  mov ah, 02h  int 21h  inc si  cmp [si], '$'  jnz L  end start | Graphical user interface, application  Description automatically generated |

## Print a string in reverse

|  |  |
| --- | --- |
| .data  name db "Hello, World!$"  .code  start:  mov ax, @data  mov ds, ax  mov si, offset name  mov cx, 0  move\_to\_end:  inc si  inc cx  cmp [si] , '$'  jne move\_to\_end  print\_from\_end:  dec si  mov dl, [si]  mov ah,02h  int 21h  loop print\_from\_end  end start |  |

## Print a string in reverse in 16-bit

|  |  |
| --- | --- |
| .data  string dw "Hello, World!$"  .code  start:  mov ax, @data  mov ds, ax  mov si, offset string  mov cx, 0  move\_to\_end:  inc si  inc cx  cmp [si] , '$'  jne move\_to\_end  print\_from\_end:  dec si  mov dl, [si]  mov ah, 02h  int 21h  loop print\_from\_end  end start |  |

## Check if a string is a palindrome or not

|  |  |
| --- | --- |
| .data  string db "racecar$"  true db "Given string is a pallindrome$"  false db "Given string is NOT a pallindrome$"  .code  start:  mov ax, @data  mov ds, ax  mov si, offset string  mov di, offset string  mov dl, offset true  mov cx, 0  move\_to\_end:  inc si  inc cx  cmp [si] , '$'  jne move\_to\_end  check:  dec si  mov bh, [si] ; copy last char to BH register  mov bl, [di] ; copy first char to BL register  inc di  cmp bh, bl  jne not\_pallindrome  loop check  mov ah, 09h  int 21h  mov ah, 0  int 21h  not\_pallindrome:  mov dl, offset false  mov ah, 09h  int 21h  end start |  |

# Experiment 3 – Arithmetic

## Add two 8-bit single digit numbers

|  |  |
| --- | --- |
| .data  num1 DB 8  num2 DB 8  .code  start:  MOV AX, data  MOV DS, AX  MOV AL, num1  ADD AL, num2 ; Adding num1 and num2  ; Calculating remainder (2nd digit) and storing it in BL  MOV AH, 00  MOV BL, 10  DIV BL  MOV BL, AH  ; If the result is single-digit, print only 1 digit  CMP AL, 00  JE print2nd  ; Printing first digit  MOV DL, AL  ADD DL, 48  MOV AH, 02  INT 21h  print2nd:  ; Printing second digit  MOV DL, BL  ADD DL, 48  MOV AH, 02  INT 21h  END start |  |

## Add two 16-bit single digit numbers

|  |  |
| --- | --- |
| .data  num1 DW 8  num2 DW 8  .code  start:  MOV AX, data  MOV DS, AX  MOV AX, num1  ADD AX, num2 ; Adding num1 and num2  ; Calculating remainder (2nd digit) and storing it in BL  MOV AH, 00  MOV BL, 10  DIV BL  MOV BL, AH  ; If the result is single-digit, print only 1 digit  CMP AL, 00  JE print2nd  ; Printing first digit  MOV DL, AL  ADD DL, 48  MOV AH, 02  INT 21h  print2nd:  ; Printing second digit  MOV DL, BL  ADD DL, 48  MOV AH, 02  INT 21h  END start |  |

## Subtract two 8-bit single digit numbers

|  |  |
| --- | --- |
| .data  a DB "Enter the first number:$"  b DB "Enter the second number:$"  c DB "Difference is:$"  .code  start:  MOV AX, data  MOV DS, AX  ; Input  MOV DX, OFFSET a  MOV AH, 09H  INT 21H  MOV AH, 01H  INT 21H  SUB AL, 30H  MOV BL, AL  ; Input  MOV DX, OFFSET B  MOV AH, 09H  INT 21H  MOV AH, 01H  INT 21H  SUB AL, 30H  MOV CL, AL  MOV AL, BL  SUB AL, CL  MOV AH, 0  AAS  MOV BL, AL  ADD BL, 30H  ; Print string  MOV DX, OFFSET C  MOV AH, 09H  INT 21H  ; Print value  MOV DL, BL  MOV AH, 02H  INT 21H  END start |  |

## Subtract two 8-bit multi digit numbers

|  |  |
| --- | --- |
| .data  a db "Enter the first 2 digit number: $"  b db "Enter the second 2 digit number: $"  c db "Difference is: $"  n db 10  .code  lea dx, a  mov ah,09h  int 21h  mov ah,01h  int 21h  sub al,30h  mov bh,al  mov ah,01h  int 21h  sub al,30h  mov bl,al  ; 2nd number  mov ah,02h  mov dl,10  int 21h  mov dl,13  int 21h  lea dx,b  mov ah,09h  int 21h  mov ah,01h  int 21h  sub al,30h  mov ch,al  mov ah,01h  int 21h  sub al,30h  mov cl,al  mov ah,02h  mov dl,10  int 21h  mov dl,13  int 21h  lea dx,c  mov ah,09h  int 21h  cmp bh,ch  JL exchange  JMP subtract  exchange:  mov ah,02h  mov dl,'-'  int 21h  xchg bx,cx  subtract:  mov ax,bx  sub al,cl  aas  sub ah,ch  aas  add ax,3030h  mov dx,ax  mov ah,02h  xchg dh,dl  int 21h  xchg dh,dl  int 21h  hlt |  |

## Multiply two 8-bit single digit numbers

|  |  |
| --- | --- |
| .data  num1 DB 5  num2 DB 2  .code  start:  MOV AX, data  MOV DS, AX  MOV AL, num1  MUL num2 ; Multiplying num1 and num2  ; Calculate remainder (2nd digit) and store it in BL  MOV AH, 00  MOV BL, 10  DIV BL  MOV BL, AH  ; If the result is single digit, print only 1 digit  CMP AL, 00  JE print2nd  ; Printing first digit  MOV DL, AL  ADD DL, 48  MOV AH, 02  INT 21h  print2nd:  ; Printing second digit  MOV DL, BL  ADD DL, 48  MOV AH, 02  INT 21h  END start |  |

## Multiply two 16-bit single digit numbers

|  |  |
| --- | --- |
| .data  num1 DW 5  num2 DW 2  .code  start:  MOV AX, data  MOV DS, AX  MOV AX, num1  MUL num2 ; Multiplying num1 and num2  ; Calculating remainder (2nd digit) and storing it in BL  MOV AH, 00  MOV BL, 10  DIV BL  MOV BL, AH  ; If the result is single-digit, print only 1 digit  CMP AL, 00  JE print2nd  ; Printing first digit  MOV DL, AL  ADD DL, 48  MOV AH, 02  INT 21h  print2nd:  ; Printing second digit  MOV DL, BL  ADD DL, 48  MOV AH, 02  INT 21h  END start |  |

## Divide two 8-bit single digit numbers

|  |  |
| --- | --- |
| .data  a db "Enter the first number:$"  b db "Enter the second number:$"  c db "Division is:$"  .code  start:  MOV AX, @data  MOV DS, AX  ; Input  MOV DX, OFFSET A  MOV AH, 09H  INT 21H  MOV AH, 01H  INT 21H  SUB AL, 30H  MOV BL, AL  ; Input  MOV DX, OFFSET B  MOV AH, 09H  INT 21H  MOV AH, 01H  INT 21H  SUB AL, 30H  MOV CL, AL  MOV AL, BL  MOV AH, 0  DIV CL  MOV BX, AX  ADD BX, 3030H  ; Print string  MOV DX, OFFSET C  MOV AH, 09H  INT 21H  ; Print value  MOV DL, BH  MOV AH, 02H  INT 21H  MOV DL, BL  MOV AH, 02H  INT 21H  END start |  |

# Experiment 5 – Integer Properties

## Check if an 8-bit number is positive or negative

|  |  |
| --- | --- |
| .data  MSG1 DW "ENTER A NUMBER:$"  MSG2 DW "NUMBER IS POSITIVE$"  MSG3 DW "NUMBER IS NEGATIVE$"  NUM1 DW 9925H  NUM2 DW 2851H  .code  START:  MOV AX, @data  MOV DS, AX  MOV DX, OFFSET MSG1  MOV AH, 09H  INT 21H  MOV AH, 01H  INT 21H  MOV BL, AL  MOV AH, 01H  INT 21H  CMP BL, '-'  JZ l  ; PRINT POSITIVE  MOV DX, OFFSET MSG2  MOV AH, 09H  INT 21H  HLT  l:  ; NEGATIVE  MOV DX, OFFSET MSG3  MOV AH, 09H  INT 21H  END START |  |

## Check if a 16-bit number is positive or negative

|  |  |
| --- | --- |
| .data  MSG1 DW "ENTER A NUMBER:$"  MSG2 DW "NUMBER IS POSITIVE$"  MSG3 DW "NUMBER IS NEGATIVE$"  NUM1 DW 9925H  NUM2 DW 2851H  .code  START:  MOV AX, @data  MOV DS, AX  MOV DX, OFFSET MSG1  MOV AH, 09H  INT 21H  MOV AH, 01H  INT 21H  MOV BL, AL  MOV AH, 01H  INT 21H  CMP BL, '-'  JZ l  ; PRINT POSITIVE  MOV DX, OFFSET MSG2  MOV AH, 09H  INT 21H  HLT  l:  ; NEGATIVE  MOV DX, OFFSET MSG3  MOV AH, 09H  INT 21H  END START |  |

## Check if an 8-bit number is odd or even

|  |  |
| --- | --- |
| .data  MSG1 DW "ENTER A NUMBER:$"  MSG2 DW "NUMBER IS EVEN$"  MSG3 DW "NUMBER IS ODD$"  .code  START:  MOV AX, @data  MOV DS, AX  MOV DX, OFFSET MSG1  MOV AH, 09H  INT 21H  MOV AH, 01H  INT 21H  MOV DX, 0H  MOV BX, 02H  DIV BX  CMP DX, 0H  JNZ LABEL  ;PRINT EVEN  MOV DX, OFFSET MSG2  MOV AH, 09H  INT 21H  HLT  LABEL:  ;ODD  MOV DX, OFFSET MSG3  MOV AH, 09H  INT 21H  END START |  |

## Check if a 16-bit number is odd or even

|  |  |
| --- | --- |
| .data  MSG1 DW "ENTER A NUMBER:$"  MSG2 DW "NUMBER IS EVEN$"  MSG3 DW "NUMBER IS ODD$"  .code  START:  MOV AX, @data  MOV DS, AX  MOV DX, OFFSET MSG1  MOV AH, 09H  INT 21H  MOV AH, 01H  INT 21H  MOV DX, 0H  MOV BX, 02H  DIV BX  CMP DX, 0H  JNZ LABEL  ;PRINT EVEN  MOV DX, OFFSET MSG2  MOV AH, 09H  INT 21H  HLT  LABEL:  ;ODD  MOV DX, OFFSET MSG3  MOV AH, 09H  INT 21H  END START |  |

# Experiment 7 – Factorial

|  |  |
| --- | --- |
| .STACK 100h  .data  a DB "enter the number: $"  .code  start:  MOV AX, @data  MOV DS, AX  ; Input  MOV DX, OFFSET a  MOV AH, 09h  INT 21h  MOV AH, 01h  INT 21h  SUB AL, 30h  MOV CH, 0  MOV CL, AL  MOV AX, 1  l:MUL CX  DEC CX  CMP CX, 0  JNE l  MOV BX, 10  MOV CL, 0  m:MOV DX, 0  DIV BX  PUSH DX  INC CL  CMP AX, 0  JNE m  n:POP DX  ADD DX, 30h  MOV AH, 02h  INT 21h  DEC CL  CMP CL, 0  JNE n  END start |  |

# Experiment 8 – Fibonacci

|  |  |
| --- | --- |
| .data  a DB "Enter the number of terms: $"  .code  start:  MOV AX, @data  MOV DS, AX  ; Input  MOV DX, OFFSET a  MOV AH, 09h  INT 21h  MOV AH, 01h  INT 21h  MOV BH, AL  MOV AH, 01h  INT 21h  MOV AH, BH  SUB AX, 3030h  AAD  MOV BH, AL  MOV DL, 32  MOV AH, 02h  INT 21h  MOV DL, 48  MOV AH, 02h  INT 21h  MOV DL, 32  MOV AH, 02h  INT 21h  MOV DL, 49  MOV AH, 02h  INT 21h  DEC BH  DEC BH  MOV CX, 01  MOV SI, 00  l:MOV DI, CX  ADD CX, SI  MOV SI, DI  MOV AX, CX  MOV DI, 10  MOV BL, 0  m:MOV DX, 0  DIV DI  ADD DX, 48  PUSH DX  INC BL  CMP AX, 0  JNE m  MOV DL, 32  MOV AH, 02h  INT 21h  p:POP DX  MOV AH, 02h  INT 21h  DEC BL  CMP BL, 0  JNE p  DEC BH  CMP BH, 0  JNE l  END start |  |