

Practice Problems

Task 1: Write a code that uses below described subroutine with different data types:

“A subroutine (entitled as: “ArraySum”) in assembly language using INVOKE directive, that takes input arguments; offset of array, length of array and type of array and display the sum on screen. Arguments should be passed using stack and it should be focused that values of GPRs should not change before and after calling of subroutine. (Sum should be calculated in 32bits, despite of the type of array).”

```
INCLUDE Irvine32.inc
ArraySum PROTO,
    ptrArray:DWORD,      ; points to the array
    szArray:DWORD,       ; array elements
    tArray:DWORD         ; array type

.data
arr DWORD 123,564,6,64896,668,4,1845,547,64,1584
arr2 WORD 123,564,6,64896,668,4,1845,547,64,1584
arr3 BYTE 12,56,45,89,56,98,78,25,36,12,53,52,96,01,32,65,59,98

.code
main PROC
    INVOKE ArraySum, ADDR arr, lengthof arr, type arr
    INVOKE ArraySum, ADDR arr2, lengthof arr2, type arr2
    INVOKE ArraySum, ADDR arr3, lengthof arr3, type arr3
exit
main ENDP

ArraySum PROC,
    ptrArray:DWORD,      ; points to the array
    szArray:DWORD,       ; array elements
    tArray:DWORD         ; array type

    PUSHAD

    MOV ECX, szArray
    MOV ESI, ptrArray
    MOV EBX, tArray
    MOV EAX, 0
    CMP ECX,0
    JE PRNT
    CMP EBX, 1
    JE L1
    CMP EBX, 2
    JE L2
    CMP EBX, 4
    JE L3
    JE PRNT
L1:
    MOVZX EDX, BYTE PTR [ESI]
    ADD EAX, EDX
    ADD ESI, 1
    LOOP L1
    JMP PRNT
L2:
    MOVZX EDI, WORD PTR [ESI]
    ADD EAX, EDI
    ADD ESI, 2
    LOOP L2
    JMP PRNT
L3:
    MOVZX EDI, DWORD PTR [ESI]
    ADD EAX, EDI
    ADD ESI, 4
    LOOP L3
    JMP PRNT
```

```
        MOVZX EDX, WORD PTR [ESI]
        ADD EAX, EDX
        ADD ESI, 2
        LOOP L2
        JMP PRNT
L3:      ADD EAX, [ESI]
        ADD ESI, 4
        LOOP L3
PRNT:    CALL WRITEDEC
        CALL CRLF
        POPAD
        RET
ArraySum ENDP
END main
```

Task 2: Write a code that uses below described subroutine with different data types:

“A subroutine described in task 1 but without using PROTO directive.”

```
INCLUDE Irvine32.inc

.data
arr DWORD 123,564,6,64896,668,4,1845,547,64,1584
arr2 WORD 123,564,6,64896,668,4,1845,547,64,1584
arr3 BYTE 12,56,45,89,56,98,78,25,36,12,53,52,96,01,32,65,59,98

.code
ArraySum PROC,
    ptrArray:DWORD,          ; points to the array
    szArray:DWORD,          ; array elements
    tArray:DWORD             ; array type

    PUSHAD
    MOV ECX, szArray
    MOV ESI, ptrArray
    MOV EBX, tArray
    MOV EAX, 0
    CMP ECX, 0
    JE PRNT
    CMP EBX, 1
    JE L1
    CMP EBX, 2
    JE L2
    CMP EBX, 4
    JE L3
    JMP PRNT
L1:
    MOVZX EDX, BYTE PTR [ESI]
    ADD EAX, EDX
    ADD ESI, 1
    LOOP L1
    JMP PRNT
L2:
    MOVZX EDX, WORD PTR [ESI]
    ADD EAX, EDX
    ADD ESI, 2
    LOOP L2
    JMP PRNT
L3:
    ADD EAX, [ESI]
    ADD ESI, 4
    LOOP L3
PRNT:
    CALL WRITEDEC
    CALL CRLF
    POPAD
    RET
ArraySum ENDP

main PROC
    INVOKE ArraySum, ADDR arr, lengthof arr, type arr
    INVOKE ArraySum, ADDR arr2, lengthof arr2, type arr2
    INVOKE ArraySum, ADDR arr3, lengthof arr3, type arr3
exit
main ENDP

END main
```

Task 3: Write a code that uses below described subroutine with different data types and display sum on screen:

“A subroutine (entitled as: “ArraySum”) in assembly language using INVOKE directive, that takes input arguments; offset of array, length of array and type of array and **return the sum**. Arguments should be passed and returned using stack and it should be focused that values of GPRs should not change before and after calling of subroutine. (Sum should be calculated in 32bits, despite of the type of array).”

```
INCLUDE Irvine32.inc
ArraySum PROTO,
    ptrArray:DWORD,          ; points to the array
    szArray:DWORD,           ; array elements
    tArray:DWORD             ; array type

.data
arr DWORD 123,564,6,64896,668,4,1845,547,64,1584
arr2 WORD 123,564,6,64896,668,4,1845,547,64,1584
arr3 BYTE 12,56,45,89,56,98,78,25,36,12,53,52,96,01,32,65,59,98

.code
main PROC
    INVOKE ArraySum, ADDR arr, lengthof arr, type arr
    MOV EAX, [ESP-24]
    CALL WRITEDEC
    CALL CRLF
    INVOKE ArraySum, ADDR arr2, lengthof arr2, type arr2
    MOV EAX, [ESP-24]
    CALL WRITEDEC
    CALL CRLF
    INVOKE ArraySum, ADDR arr3, lengthof arr3, type arr3
    MOV EAX, [ESP-24]
    CALL WRITEDEC
    CALL CRLF
exit
main ENDP

ArraySum PROC,
    ptrArray:DWORD,          ; points to the array
    szArray:DWORD,           ; array elements
    tArray:DWORD             ; array type
    LOCAL SUM:DWORD

    PUSHAD

    MOV ECX, szArray
    MOV ESI, ptrArray
    MOV EBX, tArray
    MOV EAX, 0
    CMP ECX, 0
    JE _sum:
    CMP EBX, 1
    JE L1
    CMP EBX, 2
    JE L2
    CMP EBX, 4
    JE L3
    MOV EAX, 0FFFFFFFFh
    JMP _sum
```

```

L1:      MOVZX EDX, BYTE PTR [ESI]
        ADD EAX, EDX
        ADD ESI, 1
        LOOP L1
        JMP _sum

L2:      MOVZX EDX, WORD PTR [ESI]
        ADD EAX, EDX
        ADD ESI, 2
        LOOP L2
        JMP _sum

L3:      ADD EAX, [ESI]
        ADD ESI, 4
        LOOP L3

_sum:
MOV SUM, EAX
POPAD
RET
ArraySum ENDP
END main

```

Task 4: Write a code that uses below described subroutine and display array contents using DUMPMEM:

“A subroutine (entitled as: “ArrayFill”) in assembly language using INVOKE directive, that takes input arguments; offset of array. offset of arraylength and fill the array with input from user using recursion. Array should be filled until user press ‘n’ OR ‘N’ in return of prompt asking to continue entering array. This procedure should modify array and arraylen declared in .data segment. Arguments should be passed using stack and it should be focused that values of GPRs should not change before and after calling of subroutine.”

```
INCLUDE Irvine32.inc
ArrayFill PROTO,
    ptrArray:DWORD,           ; points to the array
    ptrArraylen:DWORD,       ; points to the array elements number

.data
arr DWORD 100 DUP(?)
arrlen DWORD 0
PRM1 BYTE "Enter the ",0
PRM2 BYTE "th element of Array : ",0
PRM3 BYTE "Want to enter other element of array (y/n)?",0

.code
main PROC
    MOV ESI, OFFSET arr
    INVOKE ArrayFill, ADDR arr, ADDR arrlen
    MOV ESI, OFFSET arr
    MOV EBX, TYPE arr
    MOV ECX, arrlen
    CALL DUMPMEM
exit
main ENDP

ArrayFill PROC,
    ptrArray:DWORD,           ; points to the array
    ptrArraylen:DWORD,       ; points to the array elements number

    PUSHAD

    MOV EDX, OFFSET PRM1
    CALL WRITESTRING
    MOV ESI, ptrArraylen
    MOV EAX, [ESI]
    CALL WRITEDEC
    MOV EDX, OFFSET PRM2
    CALL WRITESTRING
    CALL READINT
    MOV EDI, ptrArray
    MOV [EDI], EAX

    INC DWORD PTR [ESI]
    ADD ptrArray,4

    MOV EDX, OFFSET PRM3
    CALL WRITESTRING
    CALL READCHAR
    CALL WRITECHAR
```

```
CALL CRLF

CMP AL, 'n'
JE _RETRN
CMP AL, 'N'
JE _RETRN

INVOKE ArrayFill, ptrArray, ptrArraylen

_RETRN:
POPAD
RET
ArrayFill ENDP
END main
```