Assembly Language Extended Exercises (Chapter 4)

1) Big Endian → Little Endian (using MOV only)

```
.data
bigEndian BYTE 12h,34h,56h,78h
littleEndian DWORD ?
.code
mov al, BYTE PTR bigEndian
mov BYTE PTR littleEndian+3, al
mov al, BYTE PTR bigEndian+1
mov BYTE PTR littleEndian+2, al
mov al, BYTE PTR bigEndian+2
mov BYTE PTR littleEndian+1, al
mov al, BYTE PTR bigEndian+3
mov BYTE PTR littleEndian, al
```

2) Exchange pairs in an array (even number of elements)

```
.data
A DWORD 1,2,3,4,5,6,7,8
.code
mov ecx, LENGTHOF A/2
xor edi, edi
PAIR_LOOP:
mov eax, DWORD PTR A[edi*8]
mov ebx, DWORD PTR A[edi*8 + TYPE A]
mov DWORD PTR A[edi*8], ebx
mov DWORD PTR A[edi*8 + TYPE A], eax
inc edi
loop PAIR_LOOP
```

3) Sum of gaps between DWORD array elements

```
.data
D DWORD 0,2,5,9,10
sum DWORD ?
.code
xor eax, eax
mov ecx, LENGTHOF D
dec ecx
xor edi, edi
GAP_LOOP:
```

```
mov ebx, DWORD PTR D[edi*4 + 4]
sub ebx, DWORD PTR D[edi*4]
add eax, ebx
inc edi
loop GAP_LOOP
mov sum, eax
```

4) Copy WORD array → DWORD array

```
.data
srcW WORD 1000h, 0ABCDh, 7, 0FFFFh
dstD DWORD LENGTHOF srcW DUP(?)
.code
xor edi, edi
mov ecx, LENGTHOF srcW
COPY_LOOP:
movzx eax, WORD PTR srcW[edi*2]
mov DWORD PTR dstD[edi*4], eax
inc edi
loop COPY_LOOP
```

5) Fibonacci sequence (first 7 values)

```
.data
Fib DWORD 7 DUP(?)
.code
mov DWORD PTR Fib[0*4], 1
mov DWORD PTR Fib[1*4], 1
mov ecx, 5
mov edi, 2
FIB_LOOP:
mov eax, DWORD PTR Fib[(edi-1)*4]
add eax, DWORD PTR Fib[(edi-2)*4]
mov DWORD PTR Fib[edi*4], eax
inc edi
loop FIB_LOOP
```

6) Reverse an integer array in place

```
.data
ARR DWORD 10,20,30,40,50
.code
mov esi, OFFSET ARR
```

```
mov edi, OFFSET ARR + SIZEOF ARR - TYPE ARR
REVERSE_LOOP:
cmp esi, edi
jae REVERSE_DONE
mov eax, DWORD PTR [esi]
mov ebx, DWORD PTR [edi]
mov DWORD PTR [esi], ebx
mov DWORD PTR [edi], eax
add esi, TYPE ARR
sub edi, TYPE ARR
jmp REVERSE_LOOP
REVERSE_DONE:
```

7) Copy string in reverse order

```
.data
source BYTE "This is the source string",0
target BYTE SIZEOF source DUP('#')
.code
mov ecx, LENGTHOF source
dec ecx
mov esi, OFFSET source
mov edi, OFFSET target
xor eax, eax
REVCOPY_LOOP:
cmp eax, ecx
jae REVCOPY_DONE
mov bl, BYTE PTR [esi + (ecx-1) - eax]
mov BYTE PTR [edi + eax], bl
inc eax
jmp REVCOPY_LOOP
REVCOPY_DONE:
mov BYTE PTR [edi + ecx], 0
```

8) Rotate DWORD array forward by one element

```
.data
R DWORD 10,20,30,40
.code
mov eax, DWORD PTR R[(LENGTHOF R - 1)*TYPE R]
mov ecx, LENGTHOF R
```

```
dec ecx
mov edi, (LENGTHOF R - 1)
SHIFT_LOOP:
mov ebx, DWORD PTR R[(edi-1)*4]
mov DWORD PTR R[edi*4], ebx
dec edi
loop SHIFT_LOOP
mov DWORD PTR R[0], eax
```