1. asm:

```
STACK 1          SEGMENT    PARA      STACK
STACK_AREA    DW         100H DUP(?)

STACK_BOTTOM  EQU        $-STACK.AREA

STACK 1          ENDS

DATA1            SEGMENT    PARA
TABLE_LEN     DW         16
TABLE                    RW    200, 300, 400, 10, 20, 1, 2037H, 386KH
                         DW    41H, 40, 42H, 50, 60
RESULT                   DB    5 DUP (?), 20H,  '$'
DATA1            ENDS

CODE1            SEGMENT    PARA
                 ASSUME     CS: CODE1 , DS: DATA1 , SS: STACK1

PRINT_WORD_DEC   PROC    FAR
                 PUSH    AX
                 PUSH    BX
                 PUSH    CX
                 PUSH    DX
```

```asm
        MOV     CX , 5
        MOV     DI , offset  RESULT + 4
        MOV     BX , 10

LP_DECIMAL :

        XOR     DX, DX
        DIV     BX
        OR      DL , 30H
        MOV     [DI] , DL
        DEC     DI
        LOOP    LP_DECIMAL
        MOV     DX , offset  RESULT
        CALL    FAR  PTR  PRINT_STRING

        POP     DI
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET

PRINT_WORD_DEC  ENDP

PRINT_BYTE      PROC   FAR
        PUSH    AX
        MOV     AH, 2
        INT     21H
        POP     AX
        RET
PRINT_BYTE      ENDP
```

```
PRINT_ STRING          PROC  FAR
                       PUSH     AX
                       PUSH MOV  AH,9
                        INT    21H
                       POP      AX
                       RET
PRINT- STRING          ENDP


PRINT_ TABLE       PROC   FAR

            MOV   CX , TABLE_LEN
            MOV   SI , OFFSET  TABLE

LP1:
            MOV   AX [SI]
            CALL      PRINT_WORD_DEC
            INC   SI
            INC   SI
            LOOP  LP1

            MOV   DX , OFFSET
            CALL  PRINT_STRING

            RET
PRINT_TABLE     ENDP
```

```
BUBBLE_SORT:        PROC

            MOV     CX , TABLE_LEN
            DEC     CX
LP2:        MOV     BX , 1
            MOV     SI , OFFSET TABLE
            PUSH    CX

LP2-1:      MOV     AX ,    [SI]
            CMP     AX ,    [SI+2]
            JBE         CONTINUE
            XCHG        AX , [SI+2]
            MOV         [SI] , AX
            MOV         BX , 0

CONTINUE:       ADD     SI , 2
                LOOP        LP2-1

                POP     CX
                DEC     CX
                CMP     BX , 1
                JZ      BUBBLE_SORT-EXIT
                JMP     SHORT LP2.
```

```
BUBBLE_SORT_EXIT:
                        RET

BUBBLE_SORT             ENDP


START:
MAIN            PROC        FAR
                MOV         AX, STACK1
                MOV         SS, AX
                MOV         SP, STACK-BOTTOM
                MOV         AX, DATA1
                MOV         DS, AX
                CALL        PRINT-TABLE
                CALL        BUBBLE_SORT
                CALL        PRINT-TABLE

EXIT:
                MOV, AX     4C00 H
                INT   21H


MAIN            ENDP
CODE1           ENDS
                END         START
```

北京航空航天大学
BEIJING UNIVERSITY OF AERONAUTICS AND ASTRONAUTICS

```asm
STACK1          SEGMENT   PARA  STACK ,
STACK_AREA        DW   100H DP(?) — — — —
STACK_BOTTOM     EQU    $-STACK_AREA
  STACK1          ENDS


DATA1          SEGMENT   PAR
RESULT         DB       5 DUP(?), 20H, '$'
NEW_LINE       DB       0DH, 0AH, '$'
DATA1          ENDS    :AT

CODE1    SEGMENT   PARA
         ASSUME  CG:CODE1, DS:DATA1, SS:STACK1


PRINT_WORD_DEC    PROC    FAR
                  PUSH    AX
                  PUSH    BX
                  PUSH    CX
                  PUSH    DX
                  PUSH    DI

                  MOV    CX, 5
                  MOV    DI, OFFSET, RESULT,+4
                  MOV    BX, 10

LP_DECIMAL:
                  XOR    DX, DX
                  DIV    BX
                  OR     DL, 30H
                  MOV    [DI], DL
```

```asm
                  CMP    AX, 0
                  JЗ     LP_ZERO
                  DEC    DI
                  LOOP   LP_DECIMAL
LP_ZERO:
                  MOV    DX, DI
                  CALL   FAR PTR PRINT_STRING
                  POP    DI
                  POP    DX
                  POP    CX
                  POP    BX
                  POP    AX
                  RET

PRINT_WORD_DEC    ENDP

PRINT_BYTE        PROC    FAR
                  PUSH    AX
                  MOV    AH, 9
                  INT    21H
                  POP    AX
                  RET

PRINT_STRING    PROC  FAR
                  PUSH    AX
                  MOV    AH, 9
                  INT    21H
                  POP    AX
                  RET
```

```asm
PRINT_STRING      ENDP
```

```
PRINT_NEWLINE    PROC    FAR          GET_LOOP_1_OUT
                 PUSH    DX
                 MOV     DX, OFFSET, NEWLINE       MOV    AX, BX
                 CALL  PRINT_STRING               POP    CX
                                                  POP    BX
                 POP   DX                         RET
                 RET                   GET_INT   ENDP

PRINT_NEWLINE     ENDP               START:
                                        MAIN    PROC    FAR
  GET_INT         PROC   FAR                MOV    AX, STACK1
                  PUSH   BX                 MOV    SS, AX
                  PUSH   CX                 MOV    SP, STACK_BOTTOM
                                            MOV    AX, DATA1
                  XOR    BX, BX
                                            MOV    DS, AX
  GETINT_LOOP1
                                            CALL  GET_INT
                  MOV   AH, 1               MOV    BX, AX
                  INT   21H                 CALL  GET_INT
                  CMP   AL, 0DH
                  JE    GET_INT_LOOP-1-OUT   MUL  BX
                  SUB   AL, 30H             CALL   PRINT_WORD_DEC
                  MOV   CX, AX              CALL   PRINT_NEWLINE
                  MOV   AX, 10           EXIT:
                  MUL   BX                   MOV   AX  4C00H
                  MOV   BX, AX                INT   21H
                  ADD   BX, CX           MAIN      ENDP
                  JMP   GETINT_LOOP_1     CODE1     ENDS
                                                   END    START
```