ALEGRIA, PRINCE HEROD S.
IT1a (11007)

prince.asm*

```
.model small
.stack 100h

.data
    welcome_msg db '=========================================', 0Dh, 0Ah
                db '                 CONVERTER ', 0Dh, 0Ah
                db '=========================================', 0Dh, 0Ah, '$'
    input_num_msg db 'Enter the number to convert: $'
    input_base_msg db 0Dh, 0Ah, 'Select input base:', 0Dh, 0Ah
                   db '1. Binary ', 0Dh, 0Ah
                   db '2. Octal ', 0Dh, 0Ah
                   db '3. Decimal ', 0Dh, 0Ah
                   db '4. Hexadecimal ', 0Dh, 0Ah
                   db 'Your choice (1-4): $'
    output_base_msg db 0Dh, 0Ah, 'Select output base:', 0Dh, 0Ah
                    db '1. Binary ', 0Dh, 0Ah
                    db '2. Octal ', 0Dh, 0Ah
                    db '3. Decimal ', 0Dh, 0Ah
                    db '4. Hexadecimal ', 0Dh, 0Ah
                    db 'Your choice (1-4): $'
    result_msg db 0Dh, 0Ah, 'Conversion result: $'
    invalid_msg db 0Dh, 0Ah, 'Invalid input! Please try again.', 0Dh, 0Ah, '$'
    line_break db 0Dh, 0Ah, '$'

    number db 20 dup(0)
    converted db 20 dup(0)
    input_base dw ?
    output_base dw ?
    decimal_value dw 0

.code
main proc
```

prince.asm*

```
main proc
    mov ax, @data
    mov ds, ax

    mov ah, 09h
    lea dx, welcome_msg
    int 21h

start_input:

    mov ah, 09h
    lea dx, input_num_msg
    int 21h

    mov si, offset number
    call read_string

    ; Get the input base
    mov ah, 09h
    lea dx, input_base_msg
    int 21h

    call read_digit
    call validate_base
    mov input_base, bx

    ; Get the output base
    mov ah, 09h
    lea dx, output_base_msg
    int 21h

    call read_digit
```

ALEGRIA, PRINCE HEROD S.
IT1a (11007)

prince.asm

```
    call validate_base
    mov output_base, bx


    mov ax, input_base
    cmp ax, output_base
    jne bases_different

    mov ah, 09h
    lea dx, result_msg
    int 21h
    mov si, offset number
    call print_string
    jmp exit_program

bases_different:

    mov si, offset number
    mov bx, input_base
    call to_decimal

    mov decimal_value, ax

    mov ax, decimal_value
    mov bx, output_base
    mov di, offset converted
    call from_decimal_no_letters

    mov ah, 09h
    lea dx, result_msg
    int 21h

```

ALEGRIA, PRINCE HEROD S.
IT1a (11007)

```
    mov si, offset converted
    call print_string

exit_program:
    ; New lines before exit
    mov ah, 09h
    lea dx, line_break
    int 21h
    int 21h

    ; Exit program
    mov ah, 4Ch
    int 21h
main endp

; Read a single digit (1-4) and convert to base
; Output: BX = base (2, 8, 10, 16)
read_digit proc
    push ax

read_again:
    mov ah, 01h
    int 21h

    cmp al, '1'
    jb invalid_digit
    cmp al, '4'
    ja invalid_digit

```

ALEGRIA, PRINCE HEROD S.
IT1a (11007)

prince.asm | GUIP.asm

```
    sub al, '0'

    mov bl, al
    mov bh, 0
    cmp bl, 1
    je binary
    cmp bl, 2
    je octal
    cmp bl, 3
    je decimal
    ; else hexadecimal

    mov bx, 16
    jmp digit_done

binary:
    mov bx, 2
    jmp digit_done

octal:
    mov bx, 8
    jmp digit_done

decimal:
    mov bx, 10

digit_done:
    pop ax
    ret

invalid digit:
```

ALEGRIA, PRINCE HEROD S.
IT1a (11007)

```
invalid_digit:
    mov ah, 09h
    lea dx, invalid_msg
    int 21h
    jmp read_again
read_digit endp

validate_base proc
    cmp bx, 2
    je base_valid
    cmp bx, 8
    je base_valid
    cmp bx, 10
    je base_valid
    cmp bx, 16
    je base_valid

    ; Invalid base
    mov ah, 09h
    lea dx, invalid_msg
    int 21h
    jmp start_input

base_valid:
    ret
validate_base endp
```

ALEGRIA, PRINCE HEROD S.
IT1a (11007)

```
validate_base endp

; Read a string from keyboard
; Input: SI = buffer address
read_string proc
    push ax
    push si

read_loop:
    mov ah, 01h
    int 21h

    cmp al, 0Dh      ; Check for Enter key
    je read_done

    mov [si], al
    inc si
    jmp read_loop

read_done:
    mov byte ptr [si], '$'

    pop si
    pop ax
    ret
read_string endp
```

ALEGRIA, PRINCE HEROD S.
IT1a (11007)

```
    pop si
    pop ax
    ret
read_string endp

to_decimal proc
    push bx
    push cx
    push dx
    push si

    xor ax, ax
    xor cx, cx

convert_loop:
    mov cl, [si]
    cmp cl, '$'
    je convert_done

    cmp cl, 'a'
    jb not_lower
    cmp cl, 'z'
    ja not_lower
    sub cl, 32        ; Convert to uppercase

not_lower:
    cmp cl, '9'
    jbe digit
    sub cl, 7         ; Adjust for A-F

digit:
    sub cl, '0'       ; Convert ASCII to digit
```

ALEGRIA, PRINCE HEROD S.
IT1a (11007)

```
    ; Check if digit is valid for the base
    cmp cl, bl
    jae invalid_digit_input

    mul bx          ; AX = AX * base
    add ax, cx      ; AX = AX + digit

    inc si
    jmp convert_loop

invalid_digit_input:
    ; Display error message
    mov ah, 09h
    lea dx, invalid_msg
    int 21h
    jmp start_input

convert_done:
    pop si
    pop dx
    pop cx
    pop bx
    ret
to_decimal endp

; Convert decimal to string in given base (without letters)
; Input: AX = decimal value, BX = base, DI = buffer address
from_decimal_no_letters proc
    push ax
```

ALEGRIA, PRINCE HEROD S.
IT1a (11007)

prince.asm | GUIP.asm

```
    push ax
    push bx
    push cx
    push dx
    push di

    xor cx, cx        ; Counter for digits

convert_loop2:
    xor dx, dx
    div bx            ; DX:AX / BX = AX remainder DX

    push dx           ; Save remainder
    inc cx

    test ax, ax
    jnz convert_loop2

    ; Pop digits in reverse order
store_loop:
    pop dx
    add dl, '0'       ; Convert to ASCII (no letters)
    mov [di], dl
    inc di

    loop store_loop

    mov byte ptr [di], '$'   ; Null-terminate

    pop di
    pop dx
    pop cx
```

ALEGRIA, PRINCE HEROD S.
IT1a (11007)

prince.asm | GUIP.asm

```
    pop bx
    pop ax
    ret
from_decimal_no_letters endp

; Print a null-terminated string
; Input: SI = string address
print_string proc
    push ax
    push dx
    push si

print_loop:
    mov dl, [si]
    cmp dl, '$'
    je print_done

    mov ah, 02h
    int 21h

    inc si
    jmp print_loop

print_done:
    pop si
    pop dx
    pop ax
    ret
print_string endp

end main
```

Output

ALEGRIA, PRINCE HEROD S.
IT1a (11007)