

ALEGRIA, PRINCE HEROD S.
IT1a (11007)

```
prince.asm* x
1 .model small
2 .stack 100h
3
4 .data
5     welcome_msg db '===== ', 0Dh, 0Ah
6                 db '                      CONVERTER ', 0Dh, 0Ah
7                 db '===== ', 0Dh, 0Ah, '$'
8     input_num_msg db 'Enter the number to convert: $'
9     input_base_msg db 0Dh, 0Ah, 'Select input base:', 0Dh, 0Ah
10                  db '1. Binary ', 0Dh, 0Ah
11                  db '2. Octal ', 0Dh, 0Ah
12                  db '3. Decimal ', 0Dh, 0Ah
13                  db '4. Hexadecimal ', 0Dh, 0Ah
14                  db 'Your choice (1-4): $'
15     output_base_msg db 0Dh, 0Ah, 'Select output base:', 0Dh, 0Ah
16                   db '1. Binary ', 0Dh, 0Ah
17                   db '2. Octal ', 0Dh, 0Ah
18                   db '3. Decimal ', 0Dh, 0Ah
19                   db '4. Hexadecimal ', 0Dh, 0Ah
20                   db 'Your choice (1-4): $'
21     result_msg db 0Dh, 0Ah, 'Conversion result: $'
22     invalid_msg db 0Dh, 0Ah, 'Invalid input! Please try again.', 0Dh, 0Ah, '$'
23     line_break db 0Dh, 0Ah, '$'
24
25     number db 20 dup(0)
26     converted db 20 dup(0)
27     input_base dw ?
28     output_base dw ?
29     decimal_value dw 0
30
31 .code
32 main proc
33     mov ax, @data
34     mov ds, ax
35
36     mov ah, 09h
37     lea dx, welcome_msg
38     int 21h
39
40     start_input:
41
42     mov ah, 09h
43     lea dx, input_num_msg
44     int 21h
45
46     mov si, offset number
47     call read_string
48
49     ; Get the input base
50     mov ah, 09h
51     lea dx, input_base_msg
52     int 21h
53
54     call read_digit
55     call validate_base
56     mov input_base, bx
57
58     ; Get the output base
59     mov ah, 09h
60     lea dx, output_base_msg
61     int 21h
62
63     call read digit
```

```
prince.asm X
64     call validate_base
65     mov output_base, bx
66
67
68     mov ax, input_base
69     cmp ax, output_base
70     jne bases_different
71
72     mov ah, 09h
73     lea dx, result_msg
74     int 21h
75     mov si, offset number
76     call print_string
77     jmp exit_program
78
79 bases_different:
80
81     mov si, offset number
82     mov bx, input_base
83     call to_decimal
84
85     mov decimal_value, ax
86
87     mov ax, decimal_value
88     mov bx, output_base
89     mov di, offset converted
90     call from_decimal_no_letters
91
92     mov ah, 09h
93     lea dx, result_msg
94     int 21h
95
```

```
95
96     mov si, offset converted
97     call print_string
98
99 exit_program:
100     ; New lines before exit
101     mov ah, 09h
102     lea dx, line_break
103     int 21h
104     int 21h
105
106     ; Exit program
107     mov ah, 4Ch
108     int 21h
109 main endp
110
111 ; Read a single digit (1-4) and convert to base
112 ; Output: BX = base (2, 8, 10, 16)
113 read_digit proc
114     push ax
115
116 read_again:
117     mov ah, 01h
118     int 21h
119
120     cmp al, '1'
121     jb invalid_digit
122     cmp al, '4'
123     ja invalid_digit
124
```

ALEGRIA, PRINCE HEROD S.
IT1a (11007)

```
prince.asm x  GUIP.asm
124
125     sub al, '0'
126
127     mov bl, al
128     mov bh, 0
129     cmp bl, 1
130     je binary
131     cmp bl, 2
132     je octal
133     cmp bl, 3
134     je decimal
135     ; else hexadecimal
136
137     mov bx, 16
138     jmp digit_done
139
140 binary:
141     mov bx, 2
142     jmp digit_done
143
144 octal:
145     mov bx, 8
146     jmp digit_done
147
148 decimal:
149     mov bx, 10
150
151 digit_done:
152     pop ax
153     ret
154
155 invalid digit:
```

```
155 invalid_digit:
156     mov ah, 09h
157     lea dx, invalid_msg
158     int 21h
159     jmp read_again
160 read_digit endp
161
162 validate_base proc
163     cmp bx, 2
164     je base_valid
165     cmp bx, 8
166     je base_valid
167     cmp bx, 10
168     je base_valid
169     cmp bx, 16
170     je base_valid
171
172     ; Invalid base
173     mov ah, 09h
174     lea dx, invalid_msg
175     int 21h
176     jmp start_input
177
178 base_valid:
179     ret
180 validate_base endp
```

```
180 validate_base endp
181
182 ; Read a string from keyboard
183 ; Input: SI = buffer address
184 read_string proc
185     push ax
186     push si
187
188 read_loop:
189     mov ah, 01h
190     int 21h
191
192     cmp al, 0Dh      ; Check for Enter key
193     je read_done
194
195     mov [si], al
196     inc si
197     jmp read_loop
198
199 read_done:
200     mov byte ptr [si], '$'
201
202     pop si
203     pop ax
204     ret
205 read_string endp
206
```

```
202     pop si
203     pop ax
204     ret
205 read_string endp
206
207 to_decimal proc
208     push bx
209     push cx
210     push dx
211     push si
212
213     xor ax, ax
214     xor cx, cx
215
216 convert_loop:
217     mov cl, [si]
218     cmp cl, '$'
219     je convert_done
220
221     cmp cl, 'a'
222     jnb not_lower
223     cmp cl, 'z'
224     ja not_lower
225     sub cl, 32      ; Convert to uppercase
226
227 not_lower:
228     cmp cl, '9'
229     jbe digit
230     sub cl, 7      ; Adjust for A-F
231
232 digit:
233     sub cl, '0'    ; Convert ASCII to digit
```

ALEGRIA, PRINCE HEROD S.

IT1a (11007)

```
234
235     ; Check if digit is valid for the base
236     cmp cl, bl
237     jae invalid_digit_input
238
239     mul bx          ; AX = AX * base
240     add ax, cx      ; AX = AX + digit
241
242     inc si
243     jmp convert_loop
244
245 invalid_digit_input:
246     ; Display error message
247     mov ah, 09h
248     lea dx, invalid_msg
249     int 21h
250     jmp start_input
251
252 convert_done:
253     pop si
254     pop dx
255     pop cx
256     pop bx
257     ret
258 to_decimal endp
259
260 ; Convert decimal to string in given base (without letters)
261 ; Input: AX = decimal value, BX = base, DI = buffer address
262 from_decimal_no_letters proc
263     push ax
```

ALEGRIA, PRINCE HEROD S.
IT1a (11007)

prince.asm	X	GUIP.asm
263		push ax
264		push bx
265		push cx
266		push dx
267		push di
268		
269		xor cx, cx ; Counter for digits
270		
271		convert_loop2:
272		xor dx, dx
273		div bx ; DX:AX / BX = AX remainder DX
274		
275		push dx ; Save remainder
276		inc cx
277		
278		test ax, ax
279		jnz convert_loop2
280		
281		; Pop digits in reverse order
282		store_loop:
283		pop dx
284		add dl, '0' ; Convert to ASCII (no letters)
285		mov [di], dl
286		inc di
287		
288		loop store_loop
289		
290		mov byte ptr [di], '\$' ; Null-terminate
291		
292		pop di
293		pop dx
294		pop cx

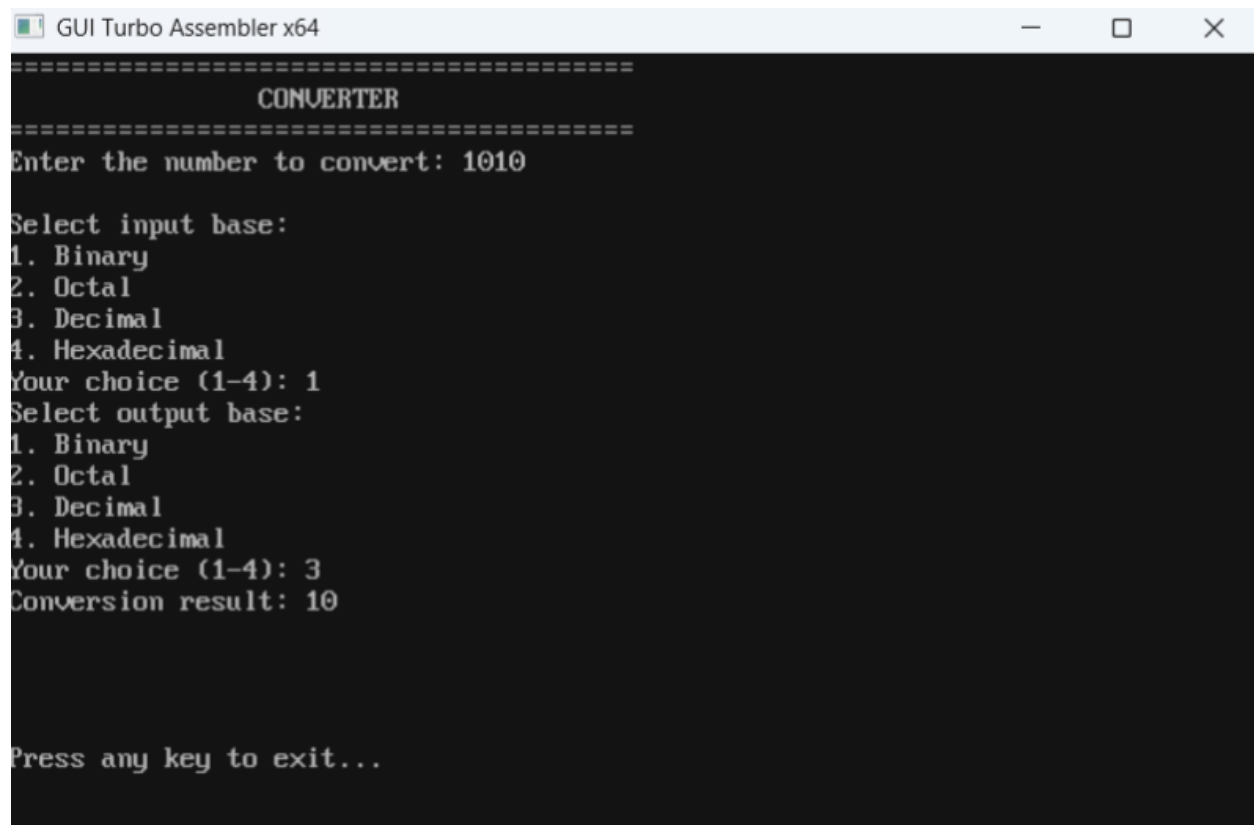
ALEGRIA, PRINCE HEROD S.

IT1a (11007)

```
prince.asm  X  GUIP.asm
294
295     pop bx
296     pop ax
297     ret
298 from_decimal_no_letters endp
299
300 ; Print a null-terminated string
301 ; Input: SI = string address
302 print_string proc
303     push ax
304     push dx
305     push si
306
307 print_loop:
308     mov dl, [si]
309     cmp dl, '$'
310     je print_done
311
312     mov ah, 02h
313     int 21h
314
315     inc si
316     jmp print_loop
317
318 print_done:
319     pop si
320     pop dx
321     pop ax
322     ret
323 print_string endp
324
325 end main
```

Output

ALEGRIA, PRINCE HEROD S.
IT1a (11007)



```
GUI Turbo Assembler x64
=====
CONVERTER
=====
Enter the number to convert: 1010

Select input base:
1. Binary
2. Octal
3. Decimal
4. Hexadecimal
Your choice (1-4): 1
Select output base:
1. Binary
2. Octal
3. Decimal
4. Hexadecimal
Your choice (1-4): 3
Conversion result: 10

Press any key to exit...
```