Code:

```
ASSUME CS: CODE, DS:DATA
DATA SEGMENT
    ASCII DW " "
DATA ENDS

CODE SEGMENT
START:MOV AX,DATA
     MOV DS,AX
     MOV AX,ASCII
     SUB AX,30H
     INT 21H
CODE ENDS
END START
```

Explanation:

ASSUME CS:CODE,DS:DATA

Tell assembler: use CODE for CS and DATA for DS.

```
DATA SEGMENT
 ASCII DW " "
DATA ENDS
```

Define data segment; ASCII is a word initialized with a space character.

```
CODE SEGMENT
START: MOV AX, DATA
```

Get address of DATA segment into AX (prepare DS).

```
MOV DS, AX
```

Load DS with that address — now data variables are accessible.

```
MOV AX, ASCII
```

Move the 16-bit value at label ASCII into AX.

```
 ADD AX, 30H
```

Add 30h to AX (intended to convert a ASCII to BCD — operates on whole AX here).

```
INT 21H
```

Call DOS interrupt 21h — undefined here because AH (DOS function) and parameter register (e.g. DL) are not set.

CODE ENDS

END START

End code segment; program entry point = START.

Code:

```
DATA SEGMENT
X DB 0B2H
DB 100 DUP (00)
DATA ENDS
CODE SEGMENT
ASSUME CS: CODE, DS: DATA
START:MOV AX, DATA
MOV DS, AX
MOV AX, 0000H
MOV BX, 0000H
MOV AL, X
MOV CL, 0AH
REPEAT:DIV CL
MOV [BX+8],AH
INC BX
MOV AH, 00H
CMP AL,CL
JMP REPEAT
MOV [BX+8], AL
INT 3H
CODE ENDS
END START
DATA SEGMENT
X DB 0B2H
DB 100 DUP (00)
```

DATA ENDS

X DB 0B2H → byte X = 0xB2 (178 decimal).

DB 100 DUP (00) → reserve 100 bytes (buffer) initialized to 0.

CODE SEGMENT

ASSUME CS: CODE, DS: DATA

START: MOV AX, DATA

MOV DS, AX

Load DATA segment base into AX, then DS = AX → DS points to data.

MOV AX, 0000H

MOV BX, 0000H

AX = 0x0000, BX = 0x0000 (BX used as byte-index into buffer).

MOV AL, X

AL = [DS:offset of X] = 0xB2 ; AX = 0x00B2.

MOV CL, 0AH

CL = 10 (divisor for decimal digit extraction).

REPEAT: DIV CL

DIV CL divides AX by CL (unsigned):

Quotient → AL, Remainder → AH.

Example 1st iter: 178 / 10 → AL = 17, AH = 8.

MOV [BX+8], AH

Store remainder (digit) at memory DS:[BX+8].

First store at offset 8, then 9, etc.

INC BX

BX = BX + 1 (move to next buffer slot).

MOV AH, 00H

Clear AH so next DIV uses AX = 0x00?? where ?? = previous AL (quotient).

CMP AL, CL

JMP REPEAT

CMP AL,CL compares quotient with 10.

BUG: JMP REPEAT is unconditional → infinite loop.

Intended: repeat only if quotient ≥ 10 (i.e., need more digits). Use conditional jump, e.g. CMP AL,CL / JAE REPEAT (unsigned) or CMP AL,10 / JAE REPEAT.

MOV [BX+8], AL

INT 3H

After loop: store final quotient (single most-significant digit) at DS:[BX+8].

INT 3H — breakpoint.