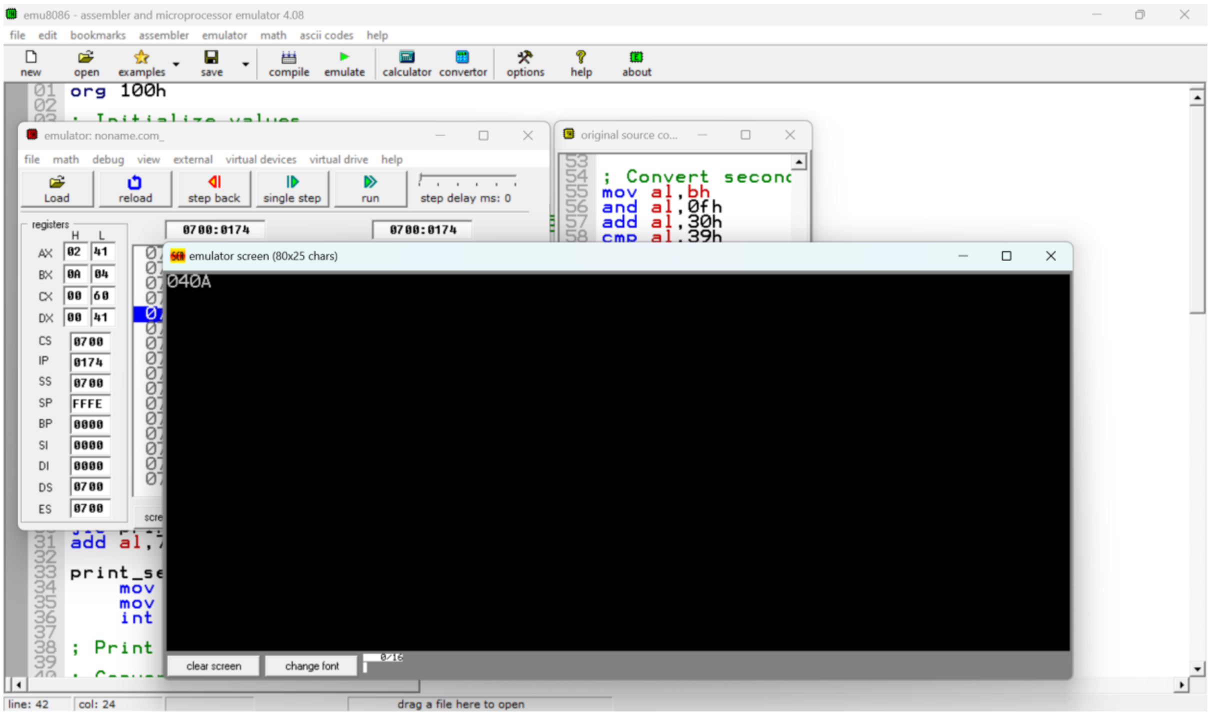
Write an assembly language program to perform division of 8-bit data.

```
org 100h
; Initialize values
mov al,56h       ; Move into AL
mov bl,13h       ; Move into BL

; Perform division (AL / BL)
idiv bl          ; AL = quotient, AH = remainder
mov bl,al        ; Store quotient in BL
mov bh,ah        ; Store remainder in BH

; Convert first digit (quotient) to ASCII
and al,0f0h      ; Mask higher nibble of AL
shr al,4         ; Shift right 4 bits to get the first hex digit
add al,30h       ; Convert to ASCII (0-9)
cmp al,39h       ; Check if it's a number or letter (0-9)
jle print_first_digit1
add al,7         ; Convert to ASCII (A-F)

print_first_digit1:
   mov dl,al     ; Move the result to DL (for printing)
   mov ah,02h    ; Print function
   int 21h       ; Interrupt to print the character

; Convert second digit (quotient) to ASCII
mov al,bl        ; Move the quotient back into AL
and al,0fh       ; Mask the lower nibble of AL
add al,30h       ; Convert to ASCII (0-9)
cmp al,39h       ; Check if it's a number or letter (0-9)
jle print_second_digit1
add al,7         ; Convert to ASCII (A-F)
```

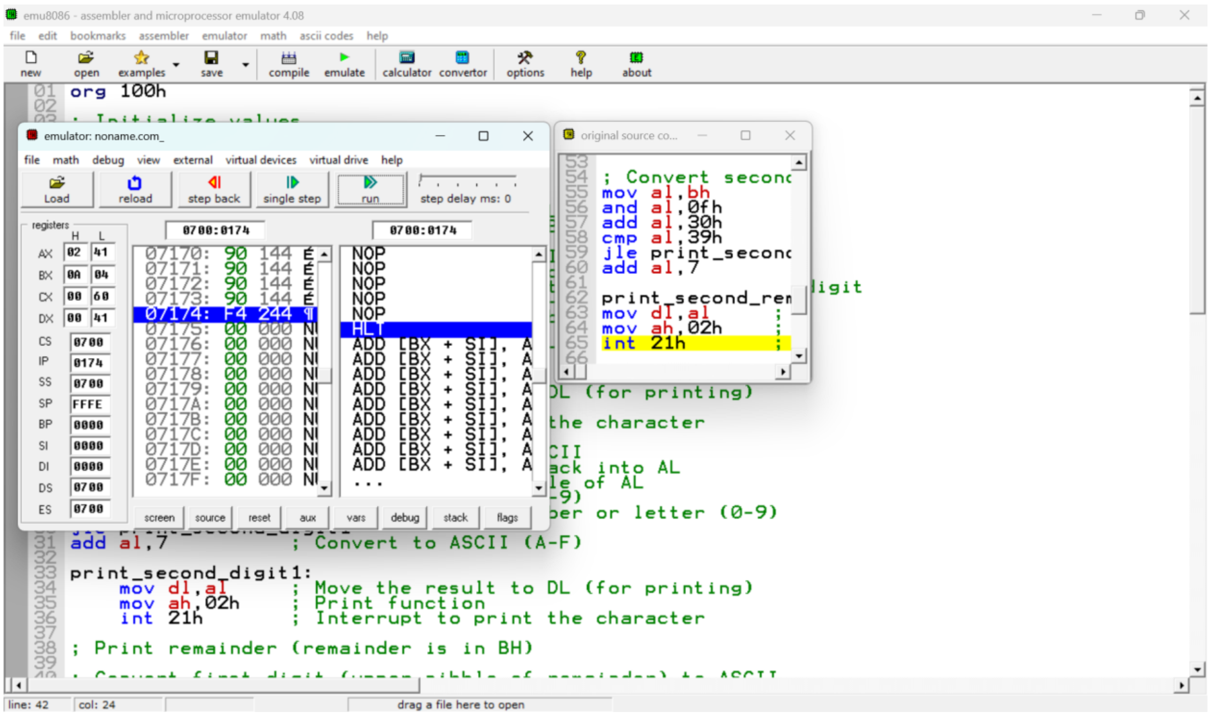
```
print_second_digit1:
    mov dl,al      ; Move the result to DL (for printing)
    mov ah,02h     ; Print function
    int 21h        ; Interrupt to print the character
; Print remainder (remainder is in BH)

; Convert first digit (upper nibble of remainder) to ASCII
mov al,bh          ; Move remainder into AL
and al,0f0h        ; Mask the higher nibble
shr al,4           ; Shift right 4 bits to get the first hex digit
add al,30h         ; Convert to ASCII (0-9)
cmp al,39h         ; Check if it's a number or letter (0-9)
jle print_first_rem_digit
add al,7           ; Convert to ASCII (A-F)

print_first_rem_digit:
    mov dl,al      ; Move the result to DL (for printing)
    mov ah,02h     ; Print function
    int 21h        ; Interrupt to print the character

; Convert second digit (lower nibble of remainder) to ASCII
mov al,bh          ; Move remainder back into AL
and al,0fh         ; Mask the lower nibble
add al,30h         ; Convert to ASCII (0-9)
cmp al,39h         ; Check if it's a number or letter (0-9)
jle print_second_rem_digit
add al,7           ; Convert to ASCII (A-F)

print_second_rem_digit:
    mov dl,al      ; Move the result to DL (for printing)
    mov ah,02h     ; Print function
    int 21h        ; Interrupt to print the character
```

OUTPUT:



```
040A
```

```
AX 02 41
BX 0A 04
CX 00 60
DX 00 41
CS 0700
IP 0174
SS 0700
SP FFFE
BP 0000
SI 0000
DI 0000
DS 0700
ES 0700
```



```
org 100h

; Convert second
mov al,bh
and al,0fh
add al,30h
cmp al,39h
jle print_second
add al,7

print_second_rem
mov dl,al
mov ah,02h
int 21h
```

```
07170: 90 144 É    NOP
07171: 90 144 É    NOP
07172: 90 144 É    NOP
07173: 90 144 É    NOP
07174: F4 244 ¶    NOP
07175: 00 000 NUL  HLT
07176: 00 000 NUL  ADD [BX + SI], A
07177: 00 000 NUL  ADD [BX + SI], A
07178: 00 000 NUL  ADD [BX + SI], A
07179: 00 000 NUL  ADD [BX + SI], A
0717A: 00 000 NUL  ADD [BX + SI], A
0717B: 00 000 NUL  ADD [BX + SI], A
0717C: 00 000 NUL  ADD [BX + SI], A
0717D: 00 000 NUL  ADD [BX + SI], A
0717E: 00 000 NUL  ADD [BX + SI], A
0717F: 00 000 NUL  ...
```

```
add al,7                ; Convert to ASCII (A-F)

print_second_digit1:
    mov dl,al           ; Move the result to DL (for printing)
    mov ah,02h          ; Print function
    int 21h             ; Interrupt to print the character

; Print remainder (remainder is in BH)
```

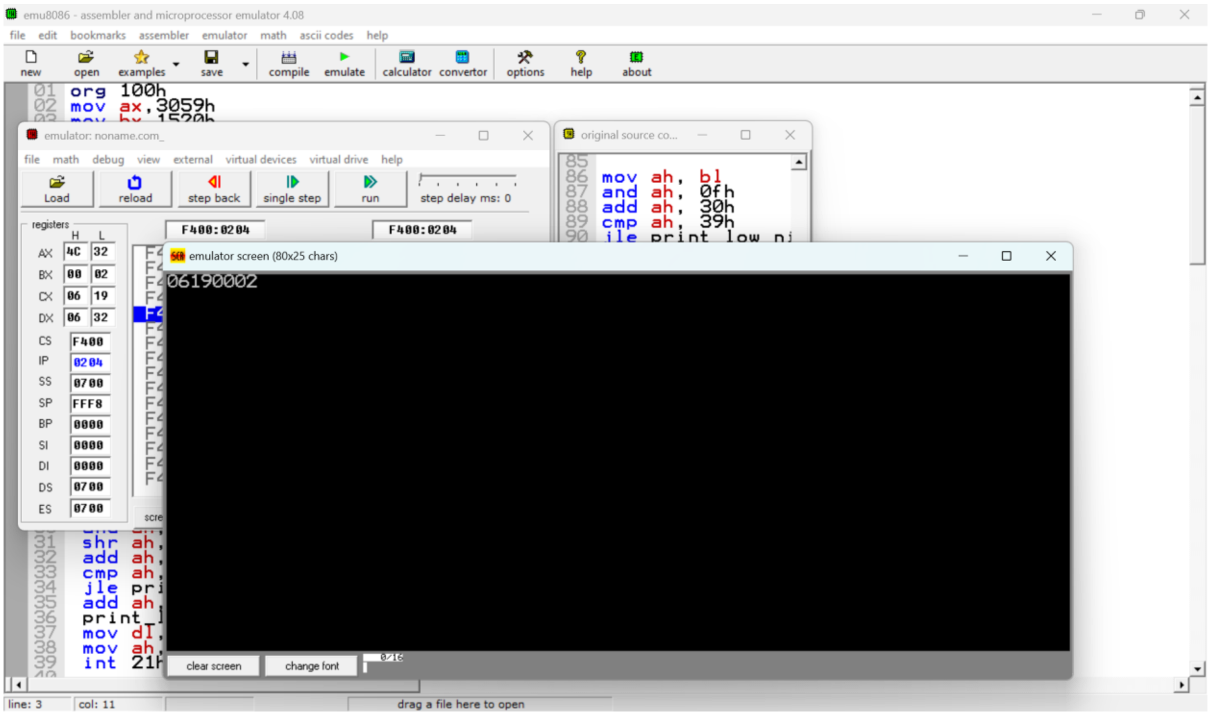
Write a program in assembly language to perform division of 16-bit data.

```
org 100h
mov ax,3059h
mov bx,1520h
div bx
 mov bx,ax
 mov cx,dx
 mov ah,ch
 and ah,0f0h
 shr ah,4
 add ah,30h
 cmp ah,39h
 jle print_high_nibble32
 add ah,7
 print_high_nibble32:
    mov dl,ah
    mov ah,02h
    int 21h
 mov ah,ch
 and ah,0fh
 add ah,30h
 cmp ah,39h
 jle print_low_nibble32
 add ah,7
 print_low_nibble32:
 mov dl,ah
 mov ah,02h
 int 21h

 mov ah,cl
 and ah,0f0h
 shr ah,4
 add ah,30h
 cmp ah,39h
 jle print_low_nibble24
 add ah,7
 print_low_nibble24:
 mov dl,ah
 mov ah,02h
 int 21h

 mov ah,cl
 and ah,0fh
 add ah,30h
 cmp ah,39h
 jle print_high_nibble24:
 add ah,7
 print_high_nibble24:
 mov dl,ah
 mov ah,02h
 int 21h


 mov ah, bh
 shr ah, 4
 add ah, 30h
 cmp ah, 39h
 jle print_high_nibble
 add ah, 7
print_high_nibble:
 mov dl, ah
 mov ah, 02h
 int 21h
 mov ah, bh
 and ah, 0fh
 add ah, 30h
 cmp ah, 39h
 jle print_low_nibble
 add ah, 7
print_low_nibble:
 mov dl, ah
 mov ah, 02h
 int 21h

 mov ah, bl
 shr ah, 4
 add ah, 30h
 cmp ah, 39h
 jle print_high_nibble2
 add ah, 7
print_high_nibble2:
 mov dl, ah
 mov ah, 02h
 int 21h

 mov ah, bl
 and ah, 0fh
 add ah, 30h
 cmp ah, 39h
 jle print_low_nibble2
 add ah, 7
print_low_nibble2:
 mov dl, ah
 mov ah, 02h
 int 21h
```

mov ah,4ch

int 21h

OUTPUT:

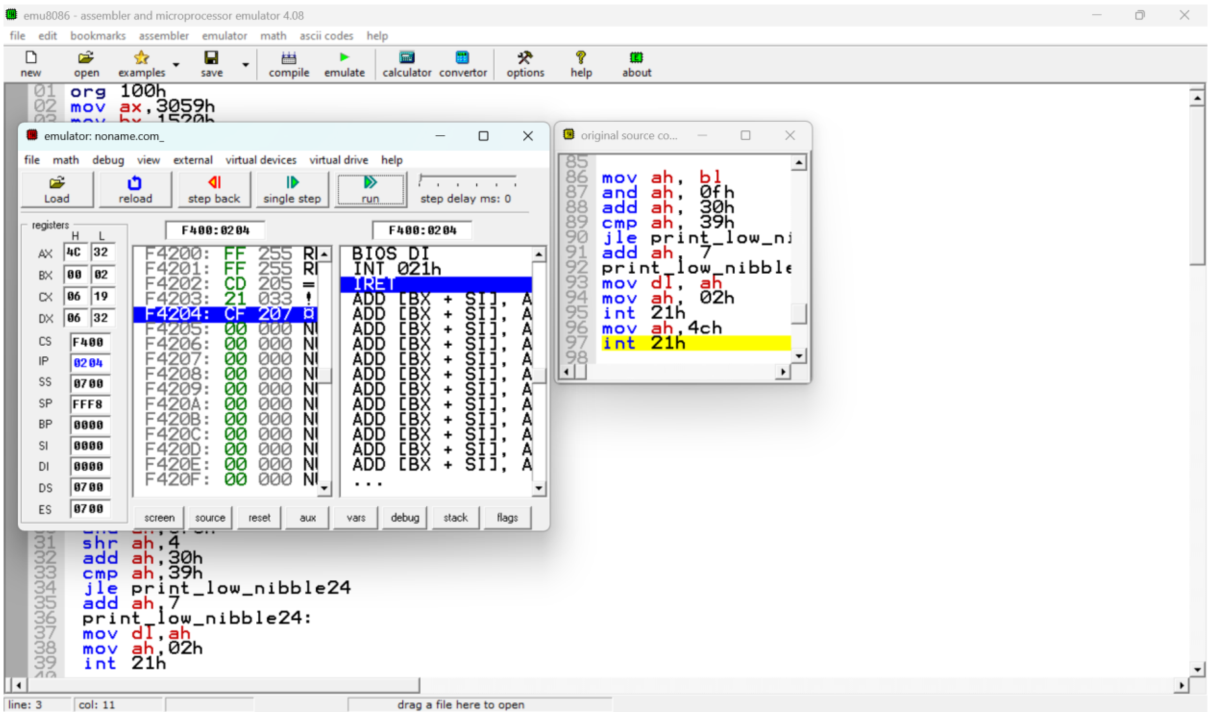


```
org 100h
mov ax,3059h
```

```
mov ah, bl
and ah, 0fh
add ah, 30h
cmp ah, 39h
jle print_low_ni
```

```
06190002
```

```
registers  H   L
AX         4C  32
BX         00  02
CX         06  19
DX         06  32
CS         F400
IP         0204
SS         0700
SP         FFF8
BP         0000
SI         0000
DI         0000
DS         0700
ES         0700
```



```
org 100h
mov ax,3059h
```

```
F400:0204                         F400:0204
F4200: FF 255 RI                  BIOS DI
F4201: FF 255 RI                  INT 021h
F4202: CD 205 =                   IRET
F4203: 21 033 !                   ADD [BX + SI], A
F4204: CF 207 ¤                   ADD [BX + SI], A
F4205: 00 000 NI                  ADD [BX + SI], A
F4206: 00 000 NI                  ADD [BX + SI], A
F4207: 00 000 NI                  ADD [BX + SI], A
F4208: 00 000 NI                  ADD [BX + SI], A
F4209: 00 000 NI                  ADD [BX + SI], A
F420A: 00 000 NI                  ADD [BX + SI], A
F420B: 00 000 NI                  ADD [BX + SI], A
F420C: 00 000 NI                  ADD [BX + SI], A
F420D: 00 000 NI                  ADD [BX + SI], A
F420E: 00 000 NI                  ADD [BX + SI], A
F420F: 00 000 NI                  ...
```

```
mov ah, bl
and ah, 0fh
add ah, 30h
cmp ah, 39h
jle print_low_ni
add ah, 7
print_low_nibble
mov dl, ah
mov ah, 02h
int 21h
mov ah,4ch
int 21h
```

```
shr ah,4
add ah,30h
cmp ah,39h
jle print_low_nibble24
add ah,7
print_low_nibble24:
mov dl,ah
mov ah,02h
int 21h
```