

计算机组成原理 Homework12 (11.18)

中国人民大学 信息学院 崔冠宇 2018202147

1. 在 200H~20FH 处存放了 8 个 word (16 位数据):

200H: 06H, 00H, 08H, 00H, 0CH, 00H, 1FH, 00H

208H: A6H, 00H, 80H, 00H, 30H, 00H, 09H, 00H

要求扫描 8 个数据, 统计其中出能被 4 整除的数据放在 210H 开始处.

解: 需要用到循环结构和选择结构.

```
1 MOV CX, 8          ; 8次循环
2 MOV SI, 200        ; 源地址
3 MOV DI, 210        ; 目标地址
4 MOV BL, 4
5 L:                 ; 循环体(10B)
6 MOV AX, [SI]       ; 读入该数
7 DIV BL
8 CMP AH, 0          ; 检验能否被4整除.
9 JNE E              ; 不能被整除, 跳转, 不执行数据传送.
10 MOV AX, [SI]
11 MOV [DI], AX       ; 可被整除, 进行数据传送.
12 ADD DI, 2
13 E:                 ; (11B)
14 ADD SI, 2
15 DEC CX
16 JNZ L
17 INT 3
```

在 DEBUG 环境中, 首先用 -e 命令将地址为 200H~20FH 的单元的数据修改为对应值, 然后用 -a 键入代码并用 -g 命令执行, 最后用 -d 观察执行效果. 部分截图如下所示:

```
DOSBox 0.74-3-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
C:\>debug
-a 200
073F:0200 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0210 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0220 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0230 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0240 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0250 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0260 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0270 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
-e 200
073F:0200 00.06 00.00 00.08 00.00 00.0C 00.00 00.1F 00.00
073F:0208 00.A6 00.00 00.80 00.00 00.30 00.00 00.09 00.00

-a 200
073F:0200 06 00 08 00 0C 00 1F 00-A6 00 80 00 30 00 09 00 .....0...
073F:0210 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0220 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0230 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0240 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0250 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0260 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0270 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
```

图 1: 设置内存单元的值

```
DOSBox 0.74-3-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
073F:010F CMP AH, 0
073F:0112 JNE 11B
073F:0114 MOV AX, [SI]
073F:0116 MOV [DI], AX
073F:0118 ADD DI, 2
073F:011B ADD SI, 2
073F:011E DEC CX
073F:011F JNZ 10B
073F:0121 INT 3
073F:0122
-g=100
AX=0102 BX=0004 CX=0000 DX=0000 SP=00FD BP=0000 SI=0210 DI=0218
DS=073F ES=073F SS=073F CS=073F IP=0121 NU UP EI PL ZR NA PE NC
073F:0121 CC INT 3
-a 200
073F:0200 06 00 08 00 0C 00 1F 00-A6 00 80 00 30 00 09 00 .....0...
073F:0210 08 00 0C 00 80 00 30 00-00 00 00 00 00 00 00 .....0.....
073F:0220 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0230 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0240 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0250 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0260 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0270 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
```

图 2: 键入代码并执行后内存单元的值

根据程序运行结果可见代码功能正确.

2. 在 200H 和 204H 分别存放了 4 个单元 BCD 码:

200H: 56H, 34H, 82H, 12H, 表示十进制数: 12823456

204H: 78H, 45H, 23H, 25H, 表示十进制数: 25234578

编程计算实现两数之和 BCD 码运算, 进位在 CY.

解: 需要用到循环结构.

```
1 MOV CX, 4          ; 4次循环
2 MOV SI, 200
3 MOV DI, 204
4 CLC                ; CF <- 0
5 L:
6 MOV AL, [SI]
7 ADC AL, [DI]
8 DAA
9 MOV [SI], AL
10 INC SI
11 INC DI
12 DEC CX
13 JNZ L
14 INT 3
```

在 DEBUG 环境中, 首先用 -e 命令将地址为 200H~207H 的单元的数据修改为对应值, 然后用 -a 键入代码并用 -g 命令执行, 最后用 -d 观察执行效果. 部分截图如下所示:

```

DOSBox 0.74-3-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
-d 200
073F:0200 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0210 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0220 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0230 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0240 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0250 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0260 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0270 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
-e 200
073F:0200 00.56 00.34 00.82 00.12

-e 204
073F:0204 00.78 00.45 00.23 00.25

-d 200
073F:0200 56 34 82 12 78 45 23 25-00 00 00 00 00 00 00 U4..xE#%.....
073F:0210 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0220 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0230 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0240 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0250 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0260 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0270 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....

```

图 3: 设置内存单元的值

```

DOSBox 0.74-3-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
073F:010A MOV AL, [SI]
073F:010C ADC AL, [DI]
073F:010E DAA
073F:010F MOV [SI], AL
073F:0111 INC SI
073F:0112 INC DI
073F:0113 DEC CX
073F:0114 JNZ 10A
073F:0116 INT 3
073F:0117
-g=100

AX=0038 BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0204 DI=0208
DS=073F ES=073F SS=073F CS=073F IP=0116 NU UP EI PL ZR NA PE NC
073F:0116 CC INT 3
-d 200
073F:0200 34 80 05 38 78 45 23 25-00 00 00 00 00 00 00 4..8xE#%.....
073F:0210 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0220 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0230 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0240 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0250 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0260 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0270 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....

```

图 4: 键入代码并执行后内存单元的值

根据程序运行结果可见代码功能正确.

3. 编写程序, 在 200H 开始的数据单元存放了一个 ASCII 码的 5 位数字, 数字大小自己设定 (不要超过 65535). 编程要求转换该数字字符串为一个 2 字节的二进制数, 结果在放在 210H~211H 中.

解: 需要用到循环结构. 约定数据 ASCII 数组按大端方式存放.

```
1 MOV CX, 5          ; 5次循环
2 MOV SI, 200
3 MOV AX, 0          ; sum = 0
4 L:
5 MOV BX, A          ; BX <- 0xA (10)
6 MUL BX             ; sum = sum * 10, (DX, AX)=AX * 0xAH
7 MOV BL, [SI]       ; BL <- char
8 SUB BL, 30          ; BL <- BL - 48, ASCII 转数字 c
9 ADD AX, BX         ; sum += c
10 INC SI
11 DEC CX
12 JNZ L
13 MOV [210], AX
14 INT 3
```

在 DEBUG 环境中, 首先用 -e 命令将地址为 200H~204H 的单元的数据修改 31H、32H、33H、34H、35H, 然后用 -a 键入代码并用 -g 命令执行, 最后用 -d 观察执行效果. 部分截图如下所示:

```
DOSBox 0.74-3-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
Z:\>C:
C:\>debug
-d 200
073F:0200 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0210 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0220 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0230 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0240 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0250 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0260 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0270 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
-e 200
073F:0200 00.31 00.32 00.33 00.34 00.35

-d 200
073F:0200 31 32 33 34 35 00 00 00-00 00 00 00 00 00 00 12345.....
073F:0210 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0220 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0230 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0240 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0250 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0260 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0270 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
```

图 5: 设置内存单元的值

```
DOSBox 0.74-3-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
073F:010C MJL BX
073F:010E MOV BL, [SI]
073F:0110 SUB BL, 30
073F:0113 ADD AX, BX
073F:0115 INC SI
073F:0116 DEC CX
073F:0117 JNZ 109
073F:0119 MOV [210], AX
073F:011C INT 3
073F:011D
-g=100
AX=3039 BX=0005 CX=0000 DX=0000 SP=00FD BP=0000 SI=0205 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=011C NU UP EI PL ZR NA PE NC
073F:011C CC INT 3
-d 200
073F:0200 31 32 33 34 35 00 00 00-00 00 00 00 00 00 00 12345.....
073F:0210 39 30 00 00 00 00 00 00-00 00 00 00 00 00 00 90.....
073F:0220 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0230 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0240 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0250 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0260 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0270 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
```

图 6: 键入代码并执行后内存单元的值

根据程序运行结果可见代码功能正确.

4. 编写程序, 在 200H 开始的数据单元存放了 2 个字节的 16 位二进制数, 编程要求将其转换为十进制数 ASCII 码表示的字符串, 结果放在 210H 起始的单元中.

解: 需要用到循环结构. 约定数据 ASCII 数组按小端方式存放.

```
1 MOV DI, 210
2 MOV AX, [200] ; num <- [200]
3 MOV BX, A ; BX <- 0xA (10)
4 L:
5 MOV DX, 0 ; DX <- 0, 32位除法准备
6 DIV BX ; c = num % 10, num /= 10
7 ADD DL, 30 ; c = c + 30H, 转成 ASCII
8 MOV [DI], DL ; [DI] <- c
9 INC DI
10 CMP AX, 0 ; num = 0, 结束
11 JNE L
12 INT 3
```

在 DEBUG 环境中, 首先用 -e 命令将地址为 200H~201H 的单元的数据修改 39H、30H, 然后用 -a 键入代码并用 -g 命令执行, 最后用 -d 观察执行效果. 部分截图如下所示:


```
DOSBox 0.74-3-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
Z:\>C:
C:\>debug
-d 200
073F:0200 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0210 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0220 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0230 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0240 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0250 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0260 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0270 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
-e 200
073F:0200 00.39 00.30

-d 200
073F:0200 39 30 00 00 00 00 00 00-00 00 00 00 00 00 00 90.....
073F:0210 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0220 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0230 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0240 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0250 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0260 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0270 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
```

图 7: 设置内存单元的值

```
DOSBox 0.74-3-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
073F:0106 MOV BX, A
073F:0109 MOV DX, 0
073F:010C DIV BX
073F:010E ADD DL, 30
073F:0111 MOV [DI], DL
073F:0113 INC DI
073F:0114 CMP AX, 0
073F:0117 JNE 109
073F:0119 INT 3
073F:011A
-g=100
AX=0000 BX=000A CX=0000 DX=0031 SP=00FD BP=0000 SI=0000 DI=0215
DS=073F ES=073F SS=073F CS=073F IP=0119 NU UP EI PL ZR NA PE NC
073F:0119 CC INT 3
-d 200
073F:0200 39 30 00 00 00 00 00 00-00 00 00 00 00 00 00 90.....
073F:0210 35 34 33 32 31 00 00 00-00 00 00 00 00 00 00 54321.....
073F:0220 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0230 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0240 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0250 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0260 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0270 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
```

图 8: 键入代码并执行后内存单元的值

根据程序运行结果可见代码功能正确.