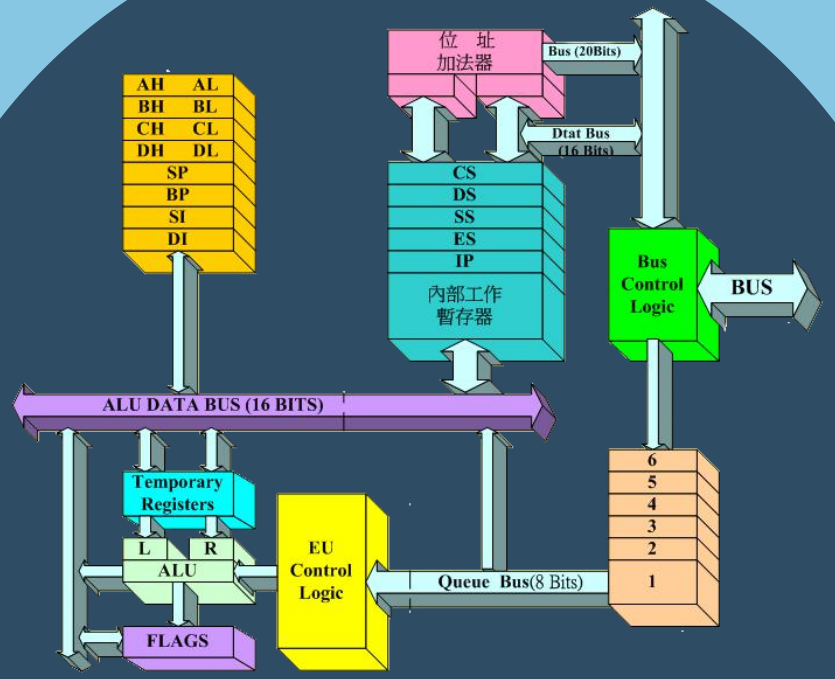
将数据、代码、栈放入不同段

贺利坚　主讲



汇编语言程序设计

Assembly Language

评价这种方案

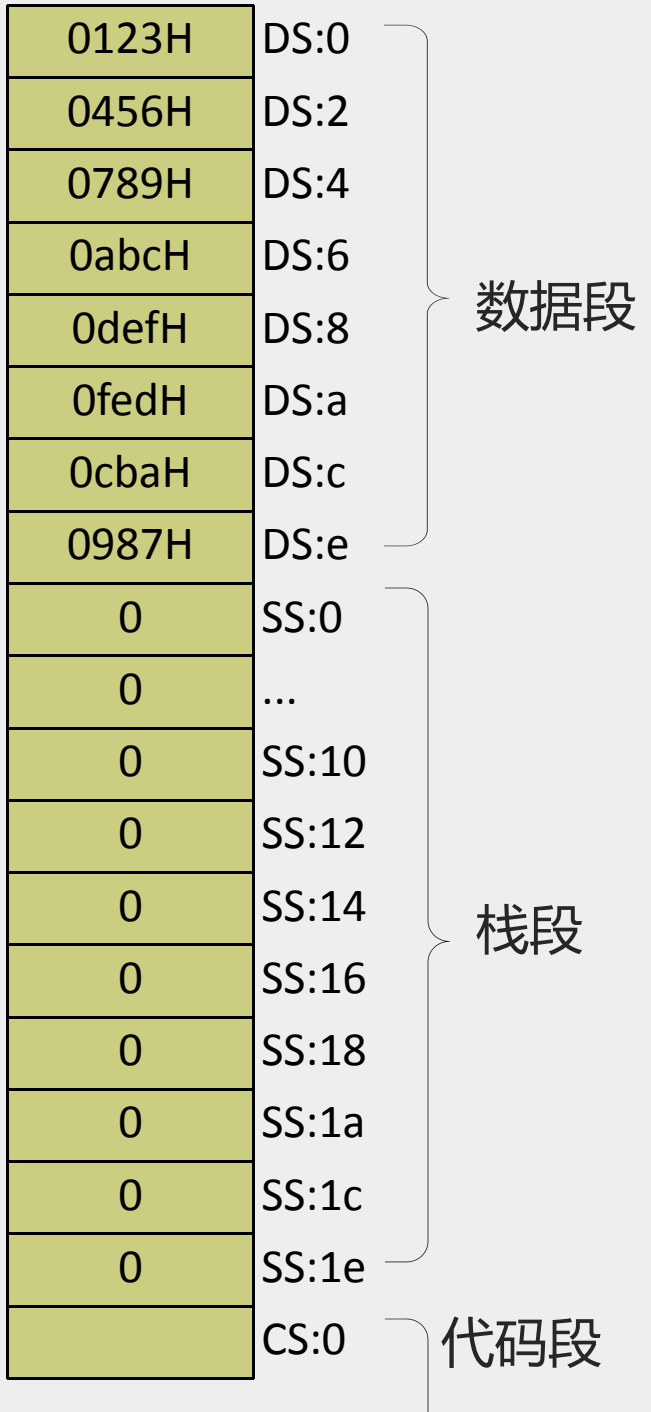
```
assume cs:codesg          ;用栈将数据逆序存放
codesg segment
        dw 0123H,0456H,0789H,0abcH,0defH,0fedH,0cbaH,0987H
        dw 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
start:  mov ax,cs
        mov ss,ax
        mov sp,30h
        ;入栈
        mov bx,0
        mov cx,8
    s:  push cs:[bx]
        add bx,2
        loop s
        ;出栈
        mov bx,0
        mov cx,8
    s0: pop cs:[bx]
        add bx,2
        loop s0
        mov ax,4c00h
        int 21h
codesg ends
end start
```

0123H	CS:0
0456H	CS:2
0789H	CS:4
0abcH	CS:6
0defH	CS:8
0fedH	CS:a
0cbaH	CS:c
0987H	CS:e
0	CS:10
0	...
0	CS:20
0	CS:22
0	CS:24
0	CS:26
0	CS:28
0	CS:2a
0	CS:2c
0	CS:2e
	CS:30

- 特点：数据、栈和代码都在一个段。
- 问题
  - 程序显得混乱，编程和阅读时都要注意何处是数据，何处是栈，何处是代码。
  - 只应用于要处理的数据很少，用到的栈空间也小，加上没有多长的代码。
- 对策：数据、栈和代码放在不同段。

将数据、代码、栈放入不同段

```
assume cs:code,ds:data,ss:stack
data segment
  dw 0123H,0456H,0789H,0abcH,0defH,0fedH,0cbaH,0987H
data ends
stack segment
  dw 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
stack ends
code segment
start:
    mov ax,stack
    mov ss,ax
    mov sp,20h
    mov ax,data
    mov ds,ax
    ; 初始化各段寄存器
    ;入栈
    mov bx,0
    mov cx,8
s:  push [bx]
    add bx,2
    loop s
    ;出栈
    mov bx,0
    mov cx,8
s0: pop [bx]
    add bx,2
    loop s0
    mov ax,4c00h
    int 21h
code ends
end start
```



在Debug中执行

```
1  assume cs:code,ds:data,ss:stack
2  data segment
3    dw 0123H,0456H,0789H,0abcH,0defH,0fedH,0cbaH,0987H
4  data ends
5  stack segment
6    dw 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
7  stack ends
8  code segment
9  start:mov ax,stack
10       mov ss,ax
11       mov sp,20h
12       mov ax,data
13       mov ds,ax
14
15       mov bx,0
16       mov cx,8
17    s:push [bx]
18       add bx,2
19       loop s
20
21       mov bx,0
22       mov cx,8
23   s0:pop [bx]
24       add bx,2
25       loop s0
26
27       mov ax,4c00h
28       int 21h
29 code ends
30 end start
```

```
C:\>debug p6-4.exe
-r
AX=FFFF  BX=0000  CX=005C  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=0769  CS=076D  IP=0000   NV UP EI PL NZ NA PO NC
076D:0000 B86B07        MOV     AX,076B
-
```

```
-u
076D:0000 B86B07        MOV     AX,076B
076D:0003 8ED0          MOV     SS,AX
076D:0005 BC2000        MOV     SP,0020
076D:0008 B86A07        MOV     AX,076A
076D:000B 8ED8          MOV     DS,AX
076D:000D BB0000        MOV     BX,0000
076D:0010 B90800        MOV     CX,0008
076D:0013 FF37          PUSH    [BX]
076D:0015 83C302        ADD     BX,+02
076D:0018 E2F9          LOOP    0013
076D:001A BB0000        MOV     BX,0000
076D:001D B90800        MOV     CX,0008
```

```
-d 076a:0 2f
076A:0000  23 01 56 04 89 07 BC 0A-EF 0D ED 0F BA 0C 87 09   #.V.............
076A:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-g

Program terminated normally
-d 076a:0 2f
076A:0000  87 09 BA 0C ED 0F EF 0D-BC 0A 89 07 56 04 23 01   ............V.#.
076A:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  87 09 BA 0C ED 0F EF 0D-BC 0A 2C 00 6D 07 12 72   ..........,.m..r
-
```