80x86 Port

Real Mode, Large Model
with Emulated Floating-Point Support

Hicks Chen

Department of Computer Science

National Chung Hsing University

Outline

- Introduction
- Development Tools
- OS_CPU.H
- OS_CPU_C.C
- OS_CPU_A.ASM

Introduction

- Running in real mode, large model.
- Use the Borland Float-Point Emulation library.
- The large model allows to reside in a 1MB memory space.
- All registers are 16-bits wide.

Development Tools

- Borland C/C++ v4.51 compiler

OS_CPU.H

```
typedef unsigned char  BOOLEAN;
typedef unsigned char  INT8U;
typedef signed   char  INT8S;
typedef unsigned int   INT16U;
typedef signed   int   INT16S;
typedef unsigned long  INT32U;
typedef signed   long  INT32S;
typedef float          FP32;
typedef double         FP64;

typedef unsigned int   OS_STK;
typedef unsigned short OS_CPU_SR;
```

OS_CPU.H

```
#define  OS_CRITICAL_METHOD    2

#if      OS_CRITICAL_METHOD == 1
#define  OS_ENTER_CRITICAL()  asm  CLI
#define  OS_EXIT_CRITICAL()   asm  STI
#endif

#if      OS_CRITICAL_METHOD == 2
#define  OS_ENTER_CRITICAL()  asm {PUSHF; CLI}
#define  OS_EXIT_CRITICAL()   asm  POPF
#endif

#if      OS_CRITICAL_METHOD == 3
#define  OS_ENTER_CRITICAL()  (cpu_sr = OSCPUSaveSR())
#define  OS_EXIT_CRITICAL()   (OSCPURestoreSR(cpu_sr))
#endif
```

OS_CPU.H

```
#define  OS_STK_GROWTH          1
#define  uCOS                   0x80
#define  OS_TASK_SW()           asm  INT   uCOS

OS_CPU_EXT  INT8U  OSTickDOSCtr;

void OSTaskStkInit_FPE_x86(OS_STK **pptos, OS_STK
                                **ppbos, INT32U *psize);
```

OS_CPU_C.C

- OSTaskStkInit()
- OSTaskCreateHook()
- OSTaskDelHook()
- OSTaskSwHook()
- OSTaskIdleHook()
- OSTaskStatHook()
- OSTimeTickHook()
- OSInitHookBegin()
- OSInitHookEnd()
- OSTCBInitHook()

OSTaskStkInit()

- Prototype：

```
OS_STK *OSTaskStkInit( void (*task)(void *pd),
                       void *pdata,
                       OS_STK *ptos,
                       INT16U opt)
```

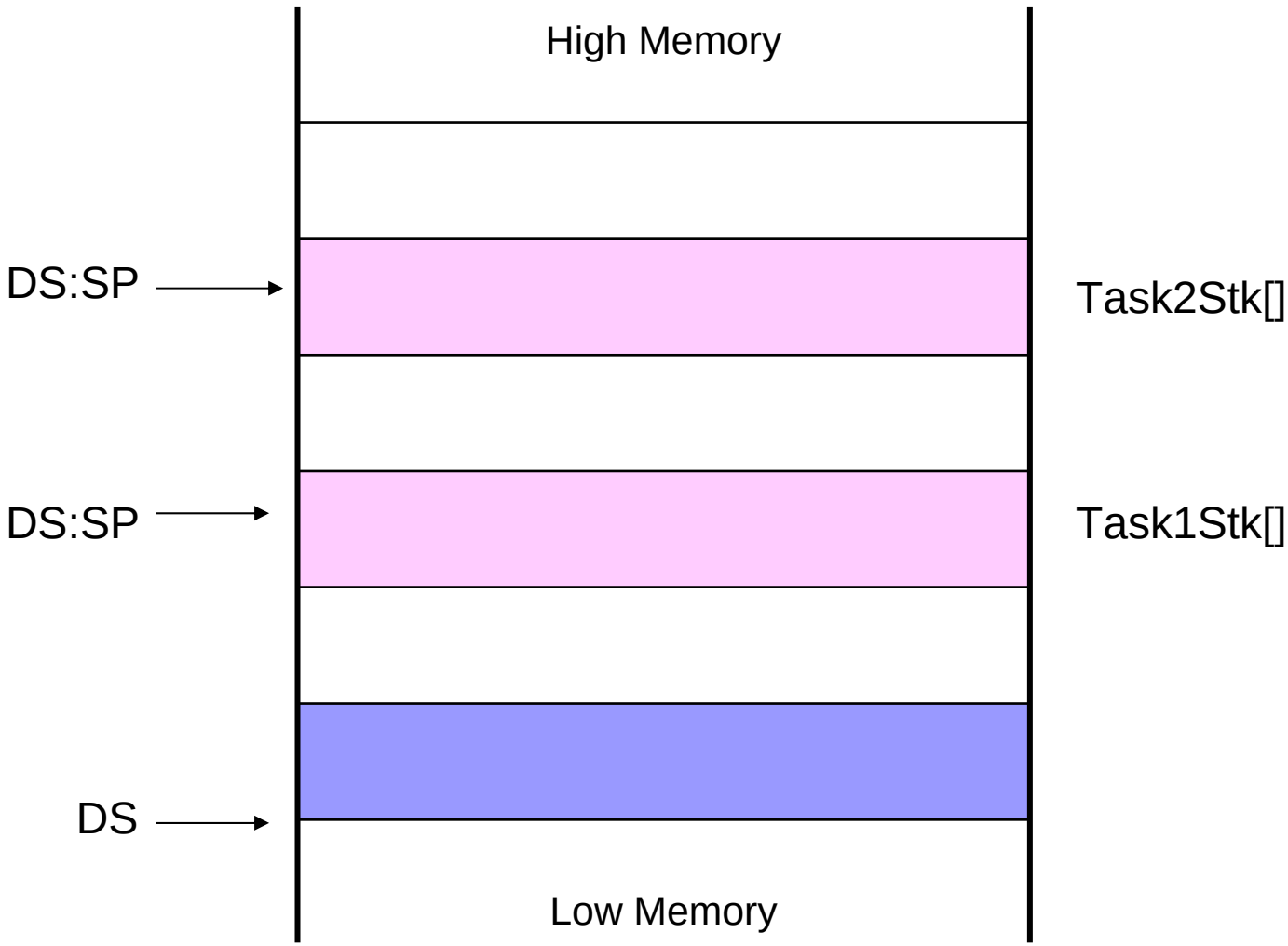
OSTaskStkInit()

```
INT16U *stk;

opt    = opt;
stk    = (INT16U *)ptos;
*stk-- = (INT16U)FP_SEG(pdata);
*stk-- = (INT16U)FP_OFF(pdata);
*stk-- = (INT16U)FP_SEG(task);
*stk-- = (INT16U)FP_OFF(task);
*stk-- = (INT16U)0x0202;
*stk-- = (INT16U)FP_SEG(task);
*stk-- = (INT16U)FP_OFF(task);
```

OSTaskStkInit()

```
*stk-- = (INT16U)0xAAAA;
*stk-- = (INT16U)0xCCCC;
*stk-- = (INT16U)0xDDDD;
*stk-- = (INT16U)0xBBBB;
*stk-- = (INT16U)0x0000;
*stk-- = (INT16U)0x1111;
*stk-- = (INT16U)0x2222;
*stk-- = (INT16U)0x3333;
*stk-- = (INT16U)0x4444;
*stk   = _DS;
return ((OS_STK *)stk);
```

OSTaskStkInit_FPE_x86()



OSTaskStkInit_FPE_x86()

- Prototype：

```
Void OStaskStkInit_FPE_x86( OS_STK **pptos,
                            OS_STK **ppbos,
                            INT32U *psize )
```

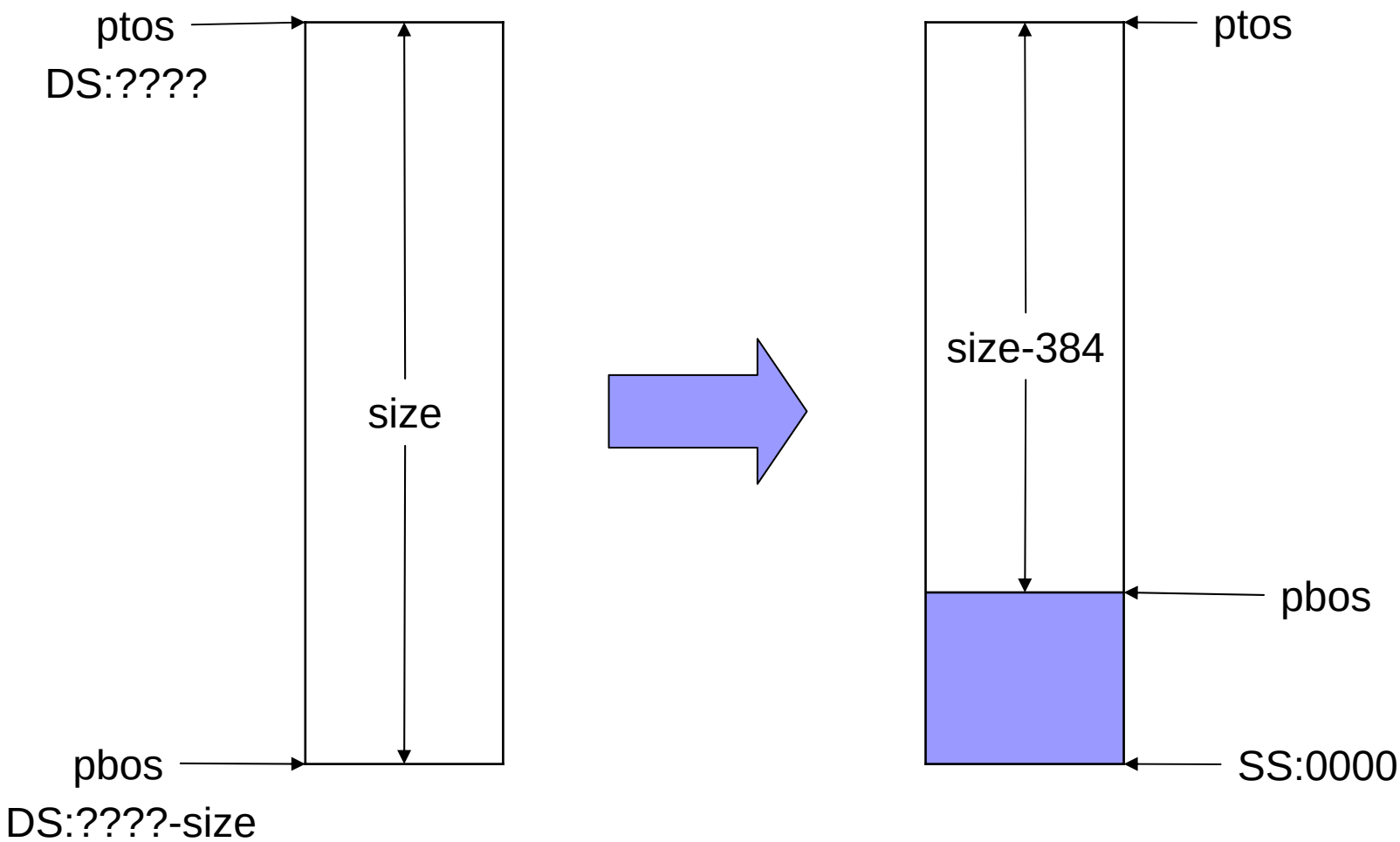
- Argument：

pptos	a pointer to the task's top-of-stack
ppbos	a pointer to the task's bottom-of-stack
psize	a pointer to a variable that contains the size of the stack.

OSTaskStkInit_FPE_x86()

```
seg       = FP_SEG(*pptos);
off       = FP_OFF(*pptos);
lin_tos   = ((INT32U)seg<<4) + (INT32U)off;
bytes     = *psize * sizeof(OS_STK);
lin_bos   = (lin_tos – bytes + 15) &0xFFFFFFF0L;

seg       = (INT16U)(lin_bos >> 4);
*ppbos    = (OS_STK *)MK_FP(seg, 0x0000);
memcpy(*ppbos, MK_FP(_SS, 0), 384);
bytes     = bytes - 16;
*pptos    = (OS_STK *)MK_FP(seg, (INT16U)bytes);
*ppbos    = (OS_STK *)MK_FP(seg, 384);
bytes     = bytes - 384;
*psize    = bytes / sizeof(OS_STK);
```

OSTaskStkInit_FPE_x86()



OS_CPU_A.ASM

- A microC/OS-II port requires that you write four assembly-language function：
  - OSStartHighRdy()
  - OSCtxSw()
  - OSIntCtxSw()
  - OSTickISR()

OSStartHighRdy()

- Description：
  - Call OSTaskSwHook() then,
  - Set OSRunning to TRUE,
  - Switch to the highest priority task.

OSStartHighRdy()

```
MOV    AX, SEG _OSTCBHighRdy              ;Reload DS
MOV    DS, AX

CALL   FAR PTR _OSTaskSwHook              ;Call user defined task
                                          ;switch hook
MOV    AL, 1                              ;OSRunning = TRUE
MOV    BYTE PTR DS:OSRunning, AL          ;Indicates that multitasking
                                          ;has started
LES    BX, DWORD PTR DS:OSTCBHighRdy      ;SS:SP =
                                          ;OSTCBHighRdy->OSTCBStkPtr
MOV    SS, ES:[BX+2]
MOV    SP, ES:[BX+0]

POP    DS                                 ;Load task's context
POP    ES
POPA

IRET                                      ;Run task
```

OSCtxSw()

- Task-level context switch
- OS_Sched() loads the address of the OC_TCB of the highest priority task into OSTCBHighRdy and then
- Executes the software-interrupt instruction by invoking the macro OS_TASK_SW()

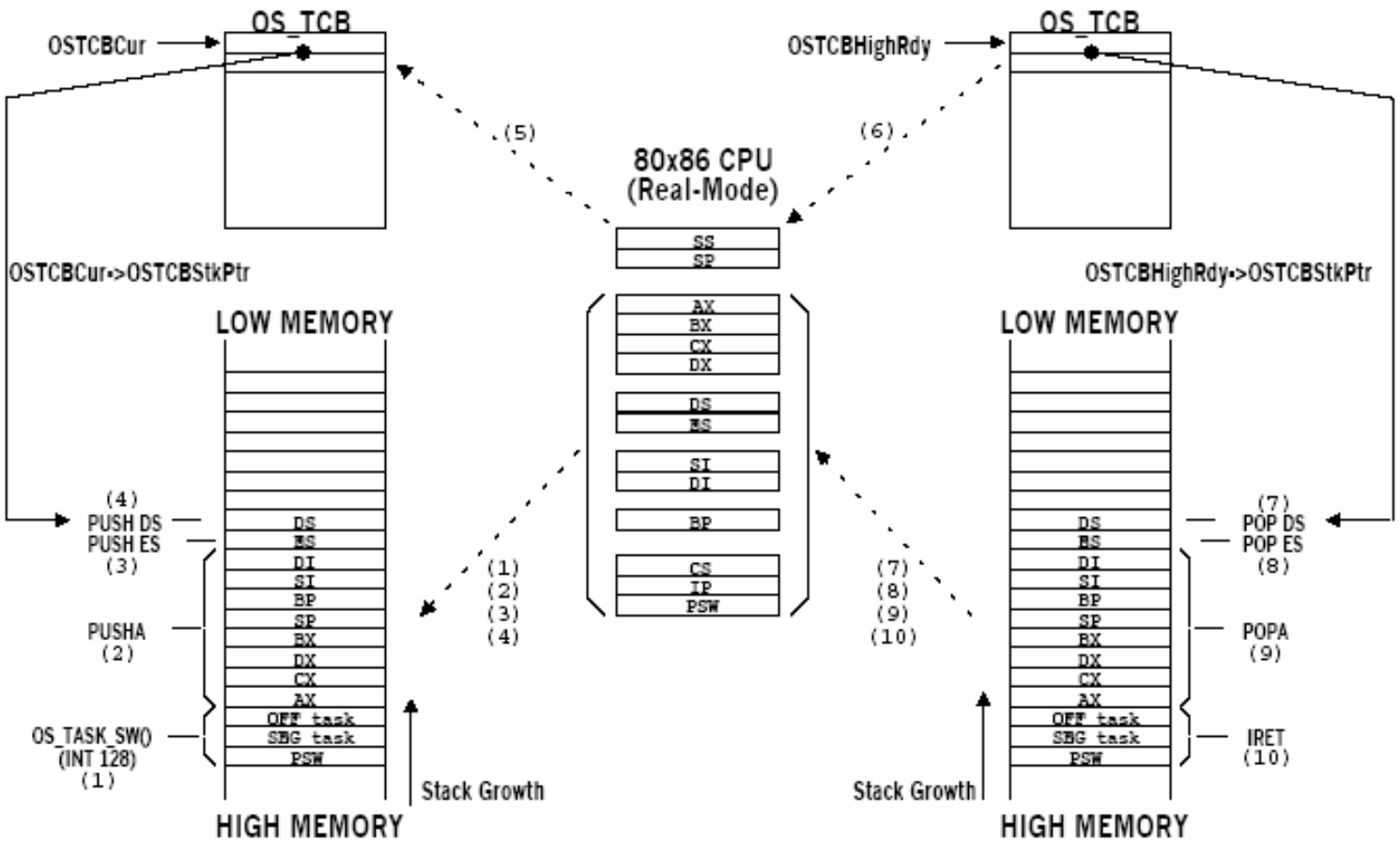
OSCtxSw()

```
PUSHA                                   ; Save current task's context
PUSH   ES
PUSH   DS

MOV    AX, SEG _OSTCBCur                ; Reload DS in case it was altered
MOV    DS, AX

LES    BX, DWORD PTR DS:_OSTCBCur       ; OSTCBCur->OSTCBStkPtr = SS:SP
MOV    ES:[BX+2], SS
MOV    ES:[BX+0], SP

CALL   FAR PTR _OSTaskSwHook            ; Call user defined task switch hook

MOV    AX, WORD PTR DS:_OSTCBHighRdy+2  ; OSTCBCur = OSTCBHighRdy
MOV    DX, WORD PTR DS:_OSTCBHighRdy
MOV    WORD PTR DS:_OSTCBCur+2, AX
MOV    WORD PTR DS:_OSTCBCur, DX
```

OSCtxSw()

```
MOV     AL, BYTE PTR DS:_OSPrioHighRdy  ; OSPrioCur = OSPrioHighRdy
MOV     BYTE PTR DS:_OSPrioCur, AL      ;

LES     BX, DWORD PTR DS:_OSTCBHighRdy  ; SS:SP = OSTCBHighRdy->OSTCBStkPtr
MOV     SS, ES:[BX+2]                   ;
MOV     SP, ES:[BX]                     ;

POP     DS                              ; Load new task's context
POP     ES                              ;
POPA                                    ;

IRET                                    ; Return to new task
```

OSCtxSw()


OS_TCB
OSTCBCur
OS_TCB
OSTCBHighRdy
(5)
(6)
80x86 CPU
(Real-Mode)
SS
SP
AX
BX
CX
DX
DS
ES
SI
DI
BP
CS
IP
PSW
OSTCBCur->OSTCBStkPtr
OSTCBHighRdy->OSTCBStkPtr
LOW MEMORY
LOW MEMORY
(4)
PUSH DS
PUSH ES
(3)
PUSHA
(2)
OS_TASK_SW()
(INT 128)
(1)
DS
ES
DI
SI
BP
SP
BX
DX
CX
AX
OFF task
SEG task
PSW
(1)
(2)
(3)
(4)
(7)
(8)
(9)
(10)
(7)
POP DS
POP ES
(8)
POPA
(9)
IRET
(10)
Stack Growth
Stack Growth
HIGH MEMORY
HIGH MEMORY

OSIntCtxSw()

- Is called by OSIntExit() to perform a context switch from an ISR.

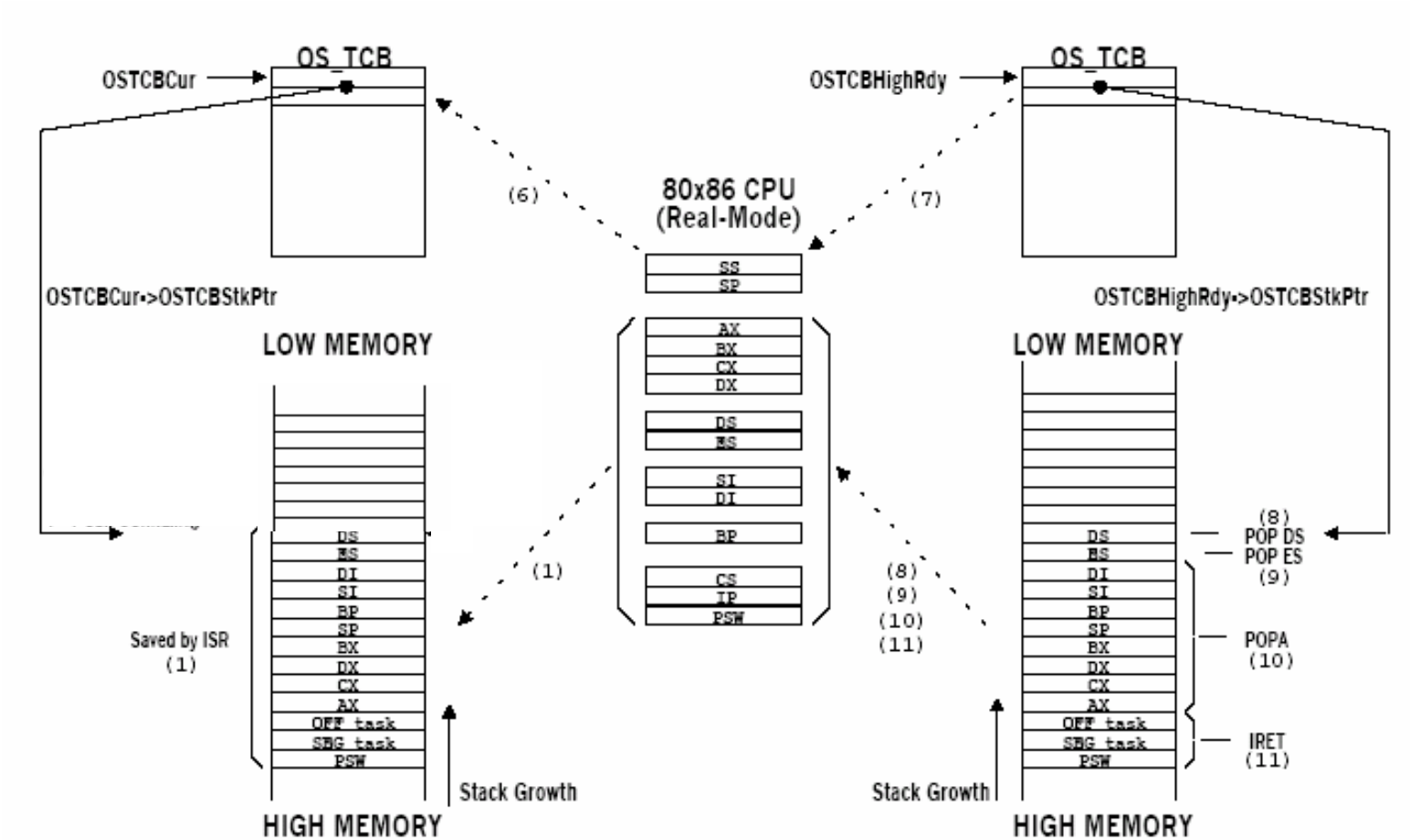
OSIntCtxSw()

```
CALL    FAR PTR _OSTaskSwHook

MOV     AX, SEG _OSTCBCur           ; Reload DS in case it was altered
MOV     DS, AX

MOV     AX, WORD PTR DS:_OSTCBHighRdy+2 ; OSTCBCur = OSTCBHighRdy
MOV     DX, WORD PTR DS:_OSTCBHighRdy
MOV     WORD PTR DS:_OSTCBCur+2, AX
MOV     WORD PTR DS:_OSTCBCur, DX
```

OSIntCtxSw()

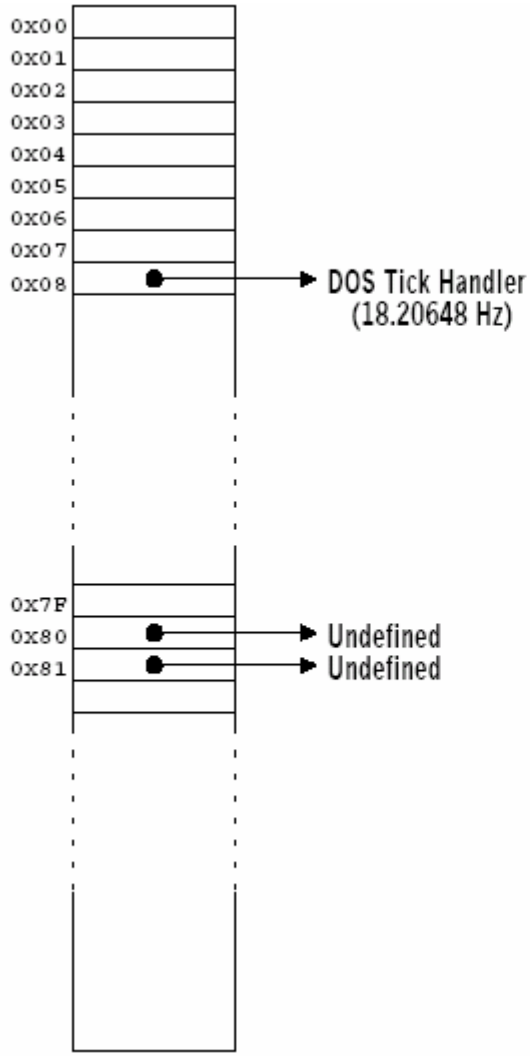
```
MOV     AL, BYTE PTR DS:_OSPrioHighRdy  ; OSPrioCur = OSPrioHighRdy
MOV     BYTE PTR DS:_OSPrioCur, AL

LES     BX, DWORD PTR DS:_OSTCBHighRdy
MOV     SS, ES:[BX+2]
MOV     SP, ES:[BX]

POP     DS                              ; Load new task's context
POP     ES
POPA

IRET                                    ; Return to new task
```

OSIntCtxSw()

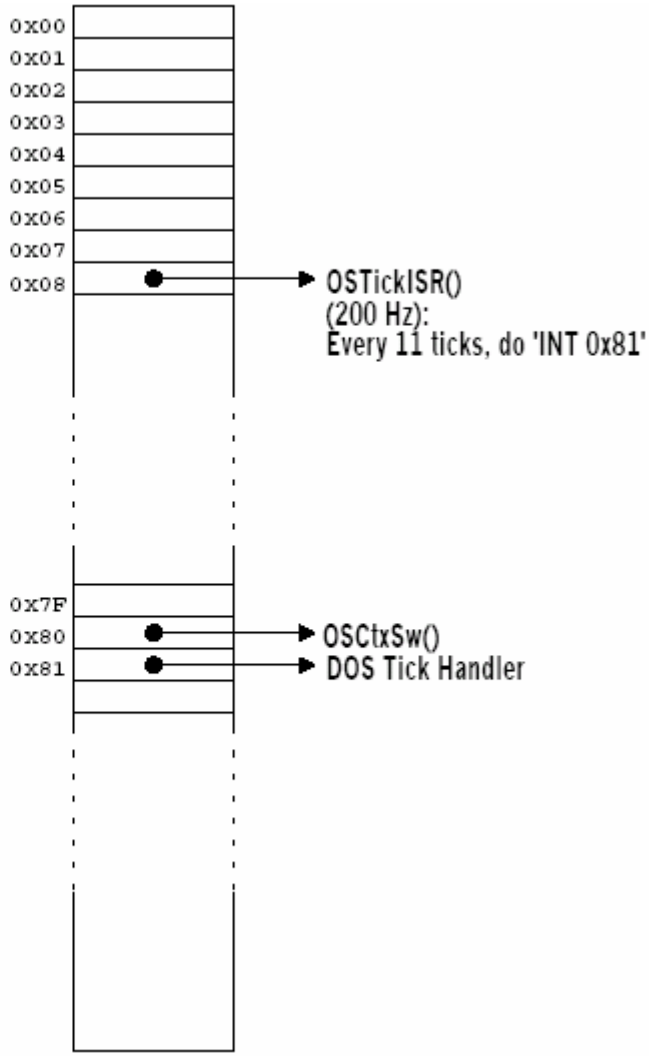

OS_TCB
OSTCBCur
OSTCBCur->OSTCBStkPtr
LOW MEMORY
Saved by ISR
(1)
DS
ES
DI
SI
BP
SP
BX
DX
CX
AX
OFF task
SEG task
PSW
Stack Growth
HIGH MEMORY
(1)
(6)
80x86 CPU
(Real-Mode)
SS
SP
AX
BX
CX
DX
DS
ES
SI
DI
BP
CS
IP
PSW
(7)
(8)
(9)
(10)
(11)
OS_TCB
OSTCBHighRdy
OSTCBHighRdy->OSTCBStkPtr
LOW MEMORY
(8)
POP DS
POP ES
(9)
POPA
(10)
IRET
(11)
Stack Growth
HIGH MEMORY

OSTickISR()

DOS Intrrupt Vector Table



microC/OSII Intrrupt Vector Table



OSTickISR()