Fundusze Europejskie
Wiedza Edukacja Rozwój

Rzeczpospolita Polska

Unia Europejska
Europejski Fundusz Społeczny

Politechnika Śląska jako Centrum Nowoczesnego Kształcenia
opartego o badania i innowacje

POWR.03.05.00-IP.08-00-PZ1/17

Projekt współfinansowany przez Unię Europejską ze środków Europejskiego Funduszu Społecznego

Microprocessor and Embedded Systems

Faculty of Automatic Control, Electronics and Computer Science, Informatics, Bachelor Degree

Lecture 6

80486
Task protection mechanisms

Bartłomiej Zieliński, PhD, DSc

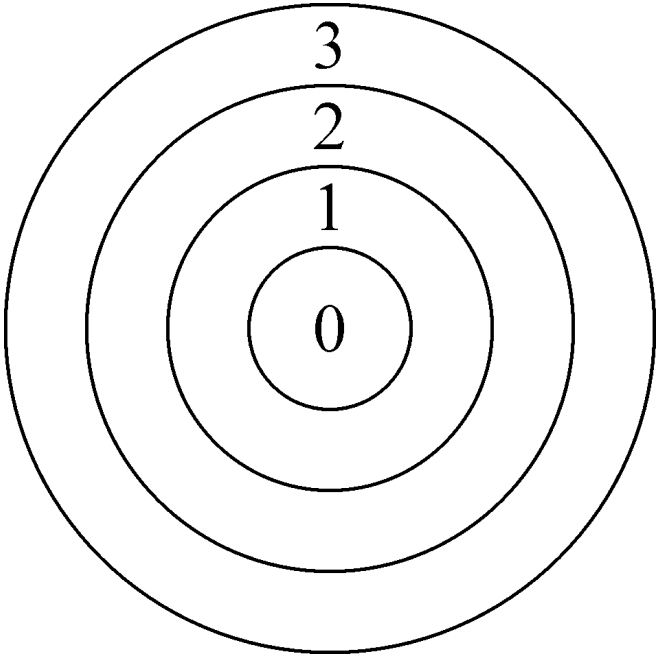
486 task protection

Program:

- Task protection model
- Task state segment
- Gate descriptors structure & application
- Exceptions & interrupts
- Cache, write buffers, burst transfers

486 task protection

- Priviledge levels
  - 0: OS kernel
  - 1: OS functions
  - 2: intermediate software
  - 3: user applications



- RPL, CPL, DPL
  - CPL can change (e.g., system function call)
  - Access to data segments
    - DPL ≥ CPL & DPL ≥ RPL

486 task protection

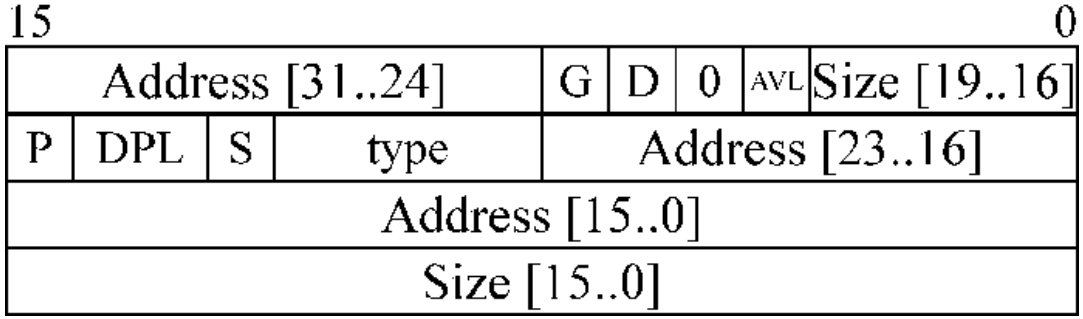
- Task state segment
  - System segment (descriptor in GDT)
    - Selector: TR
  - Contains full task context:
    - Segment registers: CS, DS, ES, FS, GS, SS
    - Addressing-related registers: LDTR, CR3
    - Working registers: EAX, EBX, ECX, EDX, ESI, EDI, ESP, EBP, EIP, Eflags
    - Stack pointers: SS:ESP also for „internal” priviledge levels

486 task protection

- Task state segment
  - Non-register fields
    - I/O access map
      - Defines which I/O addresses can be accessed by program
    - Nested Task flag, Previous task field
      - If a task is nested in another task (CALLed)
        - » e.g., interrupt service
      - Previous task points to TSS of CALLing task
    - Debug Trap flag
      - After task switching, Trap exception generated

486 task protection

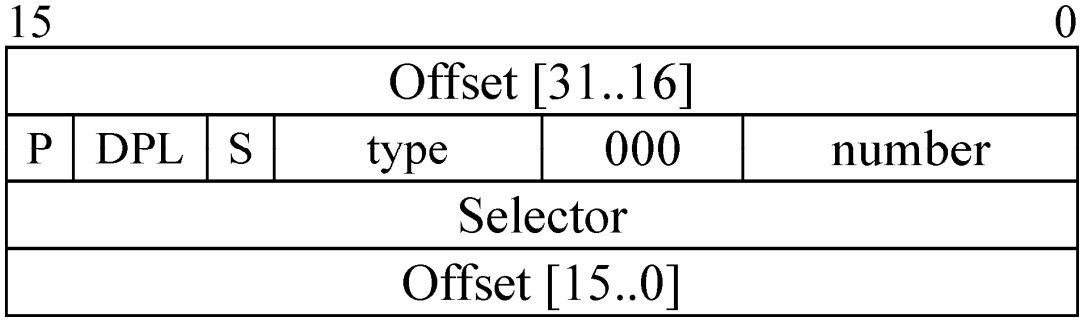
- System segment descriptor
  - E.g., TSS, LDT, etc.



  - Segment type:
    - Available TSS 286
    - Busy TSS 286
    - Available TSS 386
    - Busy TSS 386
    - LDT table
    - Call gates

486 task protection

- Call gate descriptor
  - Entry point to a code segment

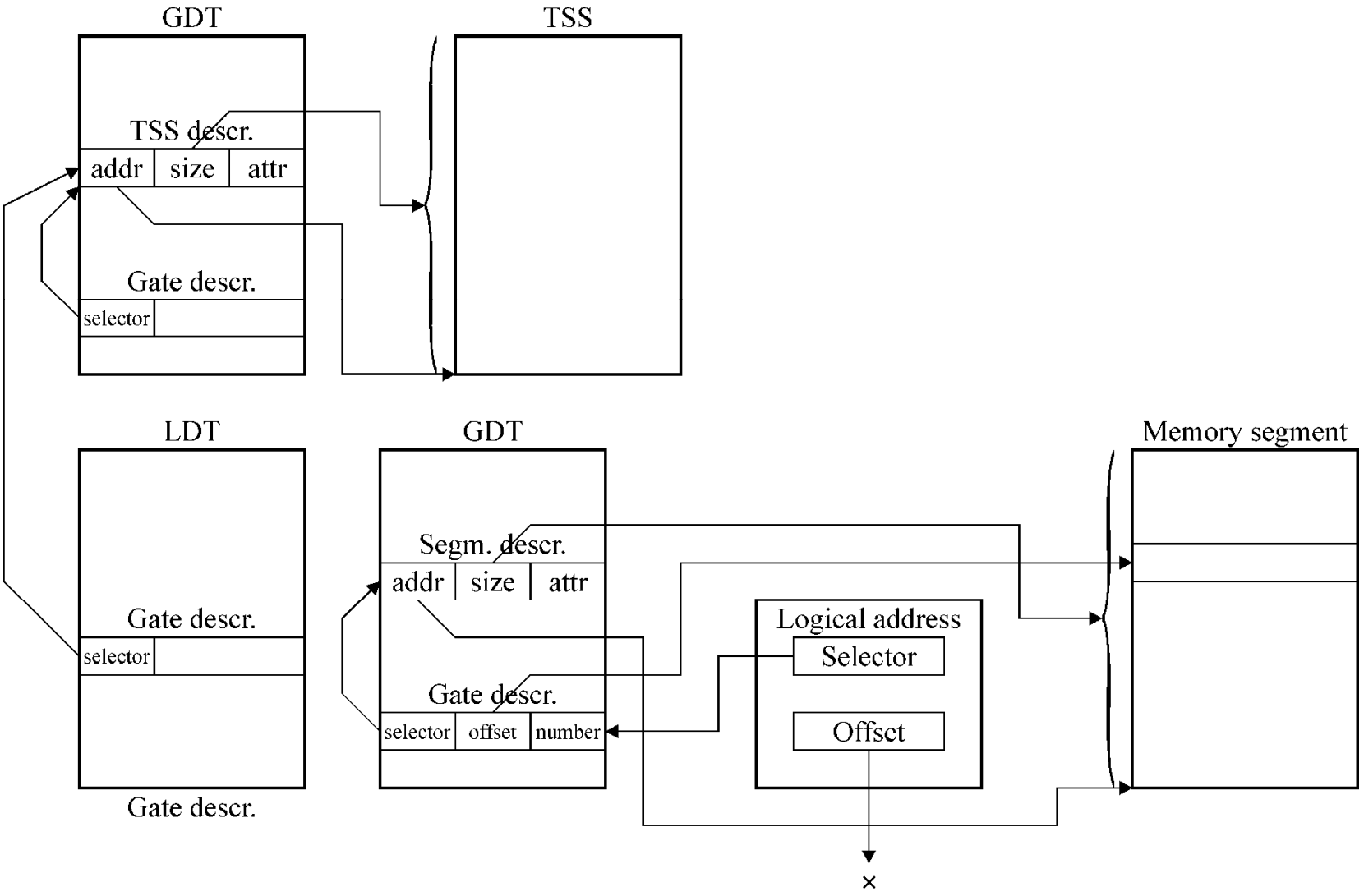


  - Segment type:
    - Call gate (286/386)
    - Task gate (only 286)
    - Interrupt gate (286/386)
    - Trap gate (286/386)

- Number used only in call gate
- Offset not used in task gate
  - Descriptor points to TSS
  - CS:EIP present in TSS

486 task protection

- Using call gate descriptors



486 task protection

- Interrupts, exceptions
  - Interrupt
    - Generated outside of the µp
      - Asynchronous to the program
      - No close relations to the current instruction
  - Exception
    - Generated inside the µp
      - Synchronous to the program
      - (Very) close relation to the current instruction

486 task protection

- Interrupts, exceptions
  - Exception classes
    - Fault (*niepowodzenie*)
      - Signalled before instruction is (fully) finished
      - Instruction can be repeated
    - Trap (*potrzask*)
      - Signalled after instruction is finished
      - Instruction can’t be repeated
    - Abort (*załamanie*)
      - Serious error
      - Instruction can’t be identified (and thus repeated)
        - » E.g., system structures incoherency

486 task protection

- Interrupts, exceptions
  - In 80486, service of both is very similar
  - Interrupt vector table
    - Real mode
      - Like in 8086, but IDTR show its location and size
      - Can be moved to any location
    - Protected mode
      - IDTR shows IDT location and size (max 256 elements)
      - Element = gate (of interrupt, trap or task)
  - Some exceptions leave error code on stack

486 task protection

- Interrupts, exceptions (1)
  - 0 (fault): divide error
  - 1 (trap): debug exceptions
  - 2 (not an exception): NMI
  - 3 (trap): breakpoint
  - 4 (trap): overflow
  - 5 (fault): bounds check
  - 6 (fault): invalid opcode
  - 7 (fault): coprocessor not available

486 task protection

- Interrupts, exceptions (2)
  - 8 (abort): double fault (exception in exception)
  - 9 (abort): coprocessor segment overrun
  - 10 (fault): invalid TSS
  - 11 (fault): segment not present
  - 12 (fault): stack exception
  - 13 (fault): general protection exception
  - 14 (fault): page fault
  - 16 (fault): coprocessor error
  - 17 (fault): alignment check interrupt

486 virtual memory

- Cache
  - Common for code and data
  - Structure
    - Logical
      - 128 sets × 4 lines × 16 B
    - Physical
      - 4 blocks × 2 KB (128 lines)
    - Tags
      - 128 × 21-b tags for each 2KB block

486 virtual memory

- Cache
  - Write Through:
    - Hit: write to cache and M
    - Miss: write to M only
  - Cache disable:
    - Hardware: $\overline{\mathrm{KEN}}$=1 (for certain addresses)
    - Software: PCD=1 (for certain pages)
    - Entirely:
      1. CD=NW=1
      2. flush

486 virtual memory

- Cache
  - Line entirely valid or invalid
  - Line can be invalidated (logically emptied)
  - Line filled for „miss read”
    - But not for „miss write”
  - Line filled
    - If invalid is found
    - If all lines valid → Least Recently Used

486 virtual memory

- Burst transfers
  - Write: max 32b if bus size=8 or 16
  - Read: max 16B (but no more than needed)
    - Address: XXXXXXX[0-F]
  - $A_{4..31}$, M/$\overline{IO}$, D/$\overline{C}$, W/$\overline{R}$ stable during transfer
  - Start: $\overline{BRDY}$ instead of $\overline{RDY}$
  - Allows to access 4 DWORDs in 5 clk's
    - 2-1-1-1 instead of 2-2-2-2

486 virtual memory

- Write buffers
  - Buffers empty, bus free → no buffering
  - Bus busy → buffering
  - Bus free → write from buffers
    - Sequence the same as during buffer filling
  - Read before write possible
    - If write „cache hit”, read „cache miss”