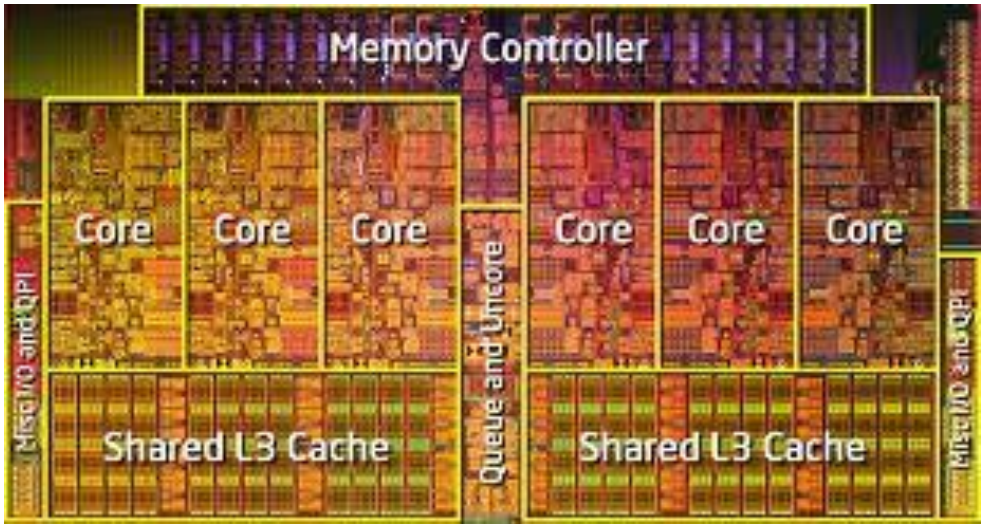
Microprocessors



Parviz Keshavarzi

Intel X86 Microprocessors

Sept, 2019

Intel 8086/8088 Microprocessors

- Intel 8086 and 8088 Microprocessors are the basis of all IBM-PC compatible computers
  (8086 introduced in 1978, first IBM-PC released in 1981)
- All Intel, AMD and other advanced microprocessors are based on and are compatible with the original 8086/8
- At Power Up and Reset time, Pentiums, Athlons etc all look like 8086 processors

Intel 8086/8088 Microprocessors

- Intel 8086 is a 16-bit microprocessor
- 16-bit data registers
- 16 or 8 bit external data bus
- Some techniques to optimise the CPU performance when it's executing programs
- Segment: Offset memory model
- Little-Endian Data Format

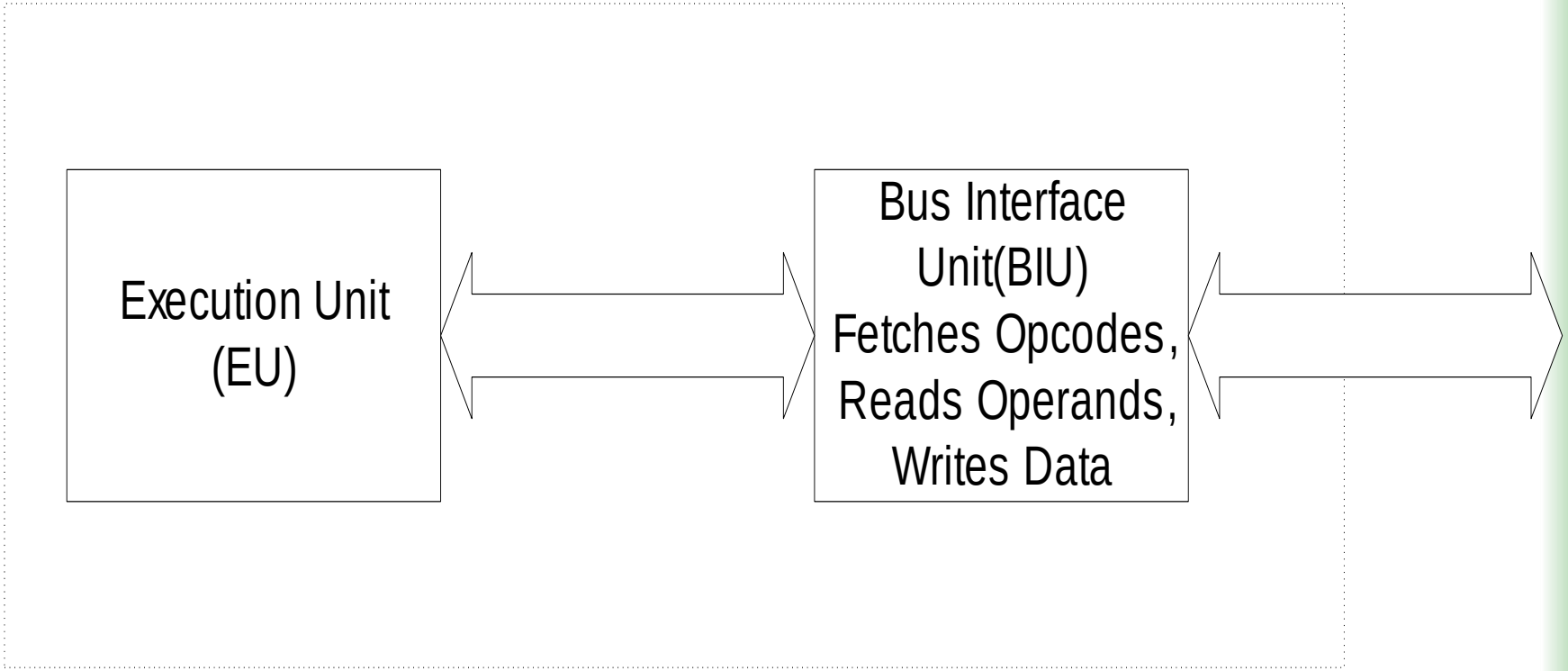
8086/8088 (1)

- Original IBM PC used 8088 micrprocessor
- 8088 is similar to the 8086 microprocessor but it has an external 8-bit bus & only 4-deep queue
  - For cost reduction reasons
- We can consider 8086 and 8088 together
- PC clones often used 8086 for better performance
- 8-bit bus reduces performance, but meant cheaper computers

8086/8088 (2)

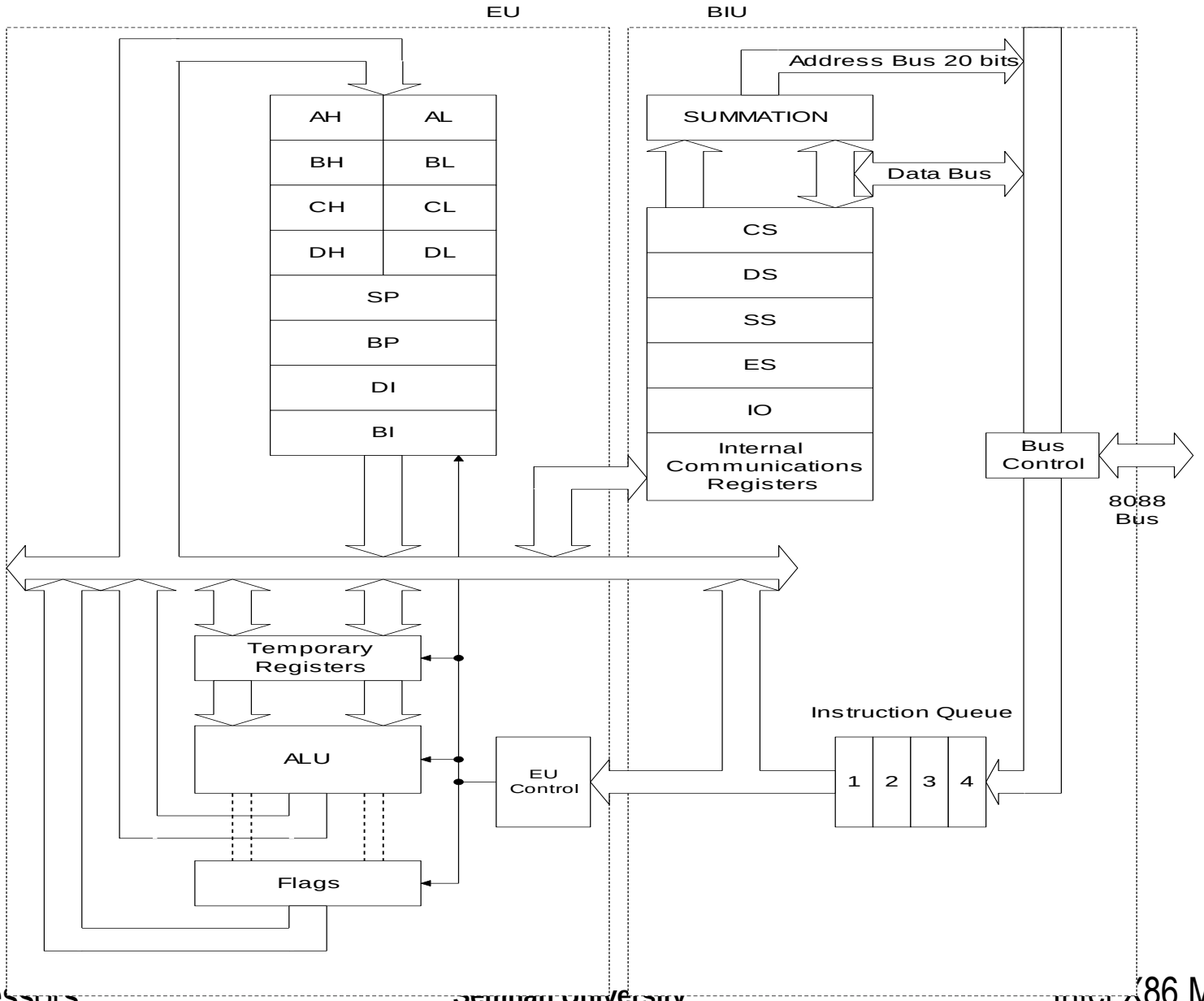
- Remember the Fetch-Decode-Execute cycle?
- Fetching from EXTERNAL MEMORY is SLOW
- The 8086/8 used an instruction queue to speed up performance
- While the processor is decoding and executing an instruction, its bus interface can be reading new instructions, since at that time the bus is not actually in use

8086/8088 Functional Units



8086/8088 MPU

8086/8088 Internal Organisation



2nd Generation Processor 286

- P2 (286) = 2nd Generation Processor
- Introduced in 1981
- CPU behind IBM AT
- Throughput of original IBM AT (6MHz) was about 500% of IBM PC (4.77MHz)
- Level of integration: 134k transistors (vs 29k in 8086)
- Still a 16-bit processor…
- Available in higher clock frequencies: 25MHz

2nd Generation Processors 286

- Fully backwards compatible to 8086
  80286 runs 8086 software without modification
- Improved instruction execution
  Average instruction takes 4.5 cycles vs. 12 cycles (8086)
- Improved instruction set
- Real mode and Protected Mode
  Multitasking-support. What happens in one area of memory doesn't affect other programs. Protected mode supported by Windows 3.0.
- 16MB addressable physical memory
- On-chip MMU (1GB virtual memory)
- Non-multiplexed address-bus and data-bus

5th Gen. Processor: Pentium

- Pentium = P5 (586) = 5th Generation Processor
  (trademarking a number designation not possible)
- Introduced: 03/1993
  (Pentium-PCs followed a few months later)
- Superscalar technology
  (2 instruction pipelines for execution of up to 2 instructions per clock cycle)
- Branch prediction
  (to avoid flushing the instruction queue and pipeline at branch-taken event)
- Internal 8kB caches for code and data
  (but external L2 cache)
- Addressbus: 32b. External Databus: 64b
  But not a 64-bit processor! Internal data paths up to 256b wide

3rd Generation Processor 386

- P3 (386) = 3rd Generation Processor
- Introduced: 10/1985
- Full 32-bit processor
  (32-bit registers. 32-bit internal and external databus. 32-bit address bus)
- 275k transistors. CMOS. 132-pin PGA package.
  (Supply current Icc=400mA. Roughly the same as 8086 !)
- Clock speeds: 16-33MHz
- P3 processors were far ahead of their time:
  It took 10 years before 32-bit operating systems became mainstream!
- First 386 PCs early 1987
  (COMPAQ)

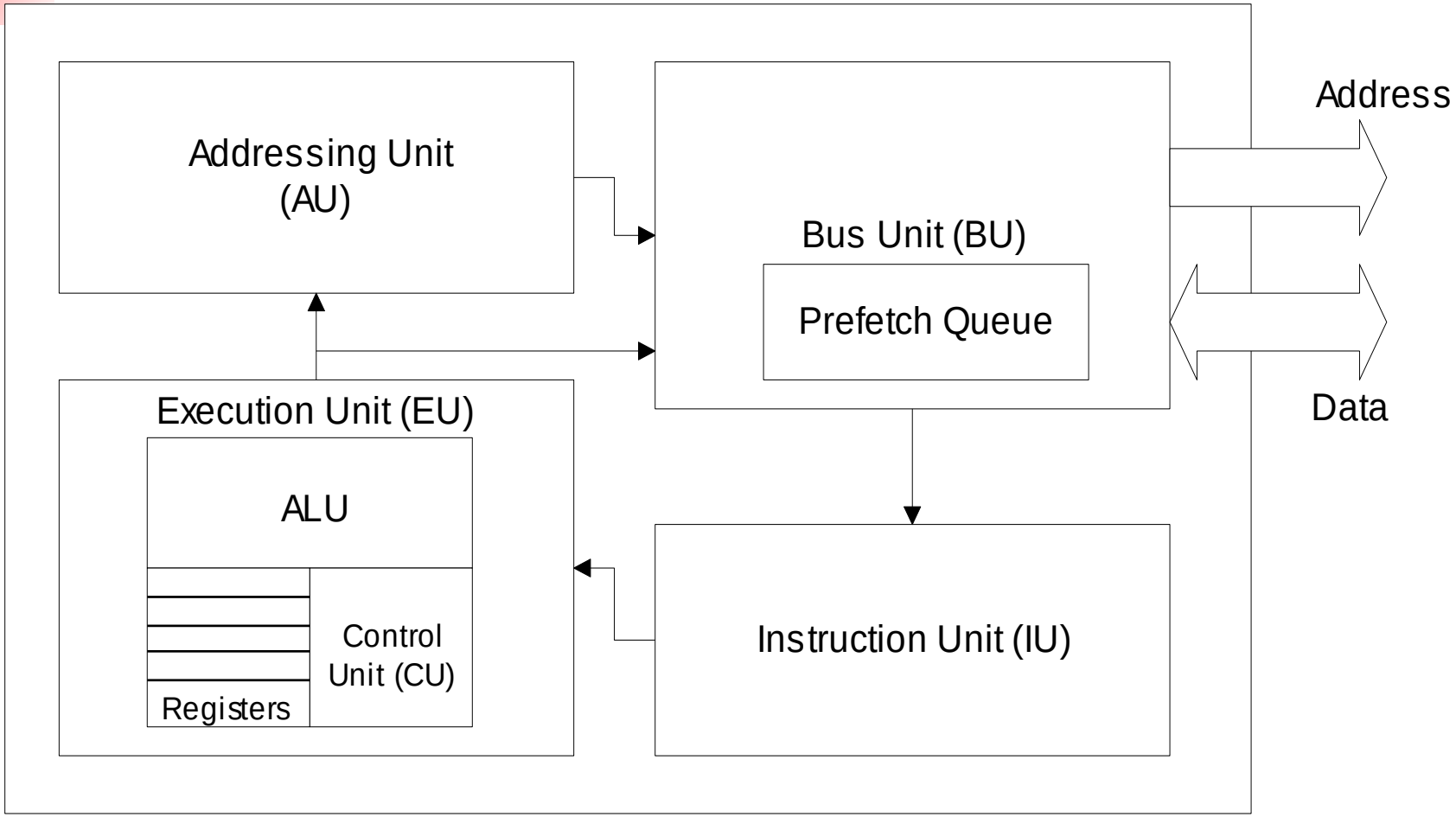
3rd Generation Processor 386

- Modes of operation:
  - Real. Protected. Virtual Real.
- Protected mode of 386 is fully compatible with 286

  Protected mode=native mode of operation. Chips are designed for advanced operating systems such as Windows NT
- New virtual real mode

  Processor can run with hardware memory protection while simulating the 8086's real-mode operation. Multiple copies of e.g. DOS can run simultaneously, each in a protected area of memory. If a program in one memory area crashes, the rest of the system is protected.

Intel 32-bit Architecture:IA-32



The 80386 includes a Bus Interface Unit for reading and providing data and instructions, witha Prefetch Queue, an IU for controlling the EU with its registers, as well as an AU for generating memory and I/O addresses

80386 Features

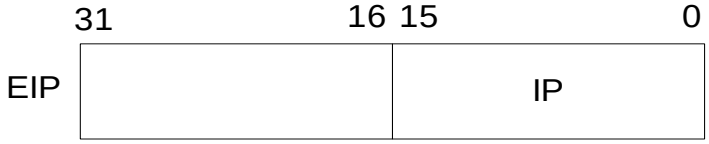
- 32-bit general and offset registers
- 16-byte prefetch queue
- Memory management unit with segmentation unit and paging unit
- 32-bit address and data bus
- 4-Gbyte physical address space
- 64-Tbyte virtual address space
- i387 numerical coprocessor
- Implementation of real, protected and virtual 8086 modes

80386 Operating Modes

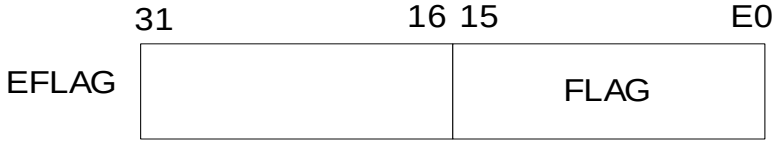
- Protected Mode for Multitasking support
- Real Mode (native 8086 mode)
  - Processor powers up in Real Mode
- System Management Mode
  - Power management or system security
  - Processor switches to separate address space, while saving the entire context of the currently running program or task

80386 Register Set

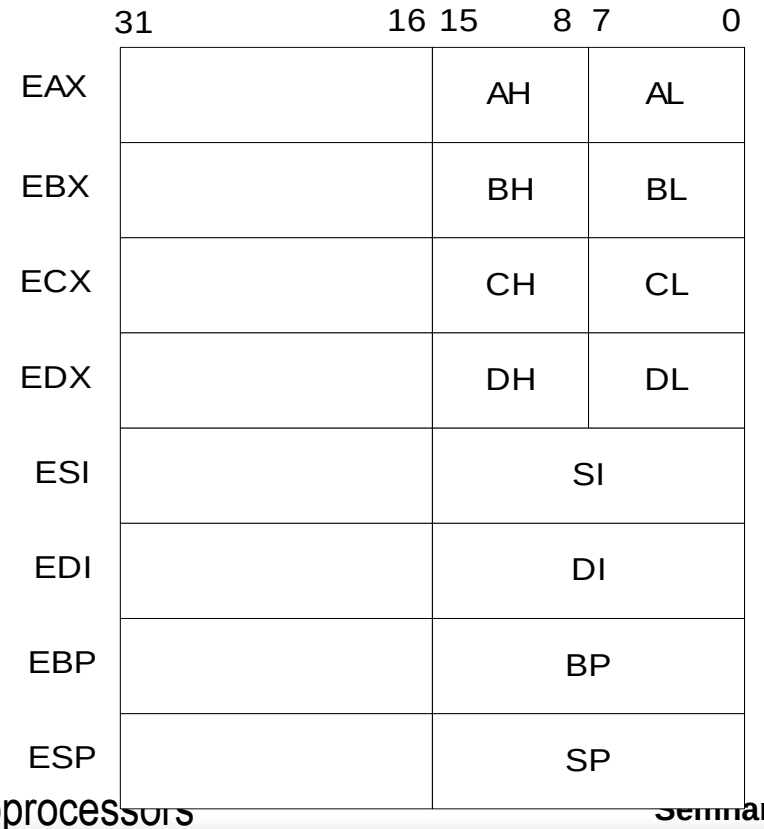
Instruction Pointer



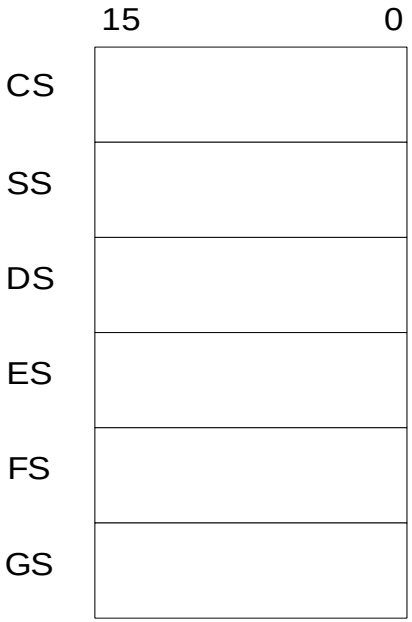
EFLAG Register



General-Purpose Registers



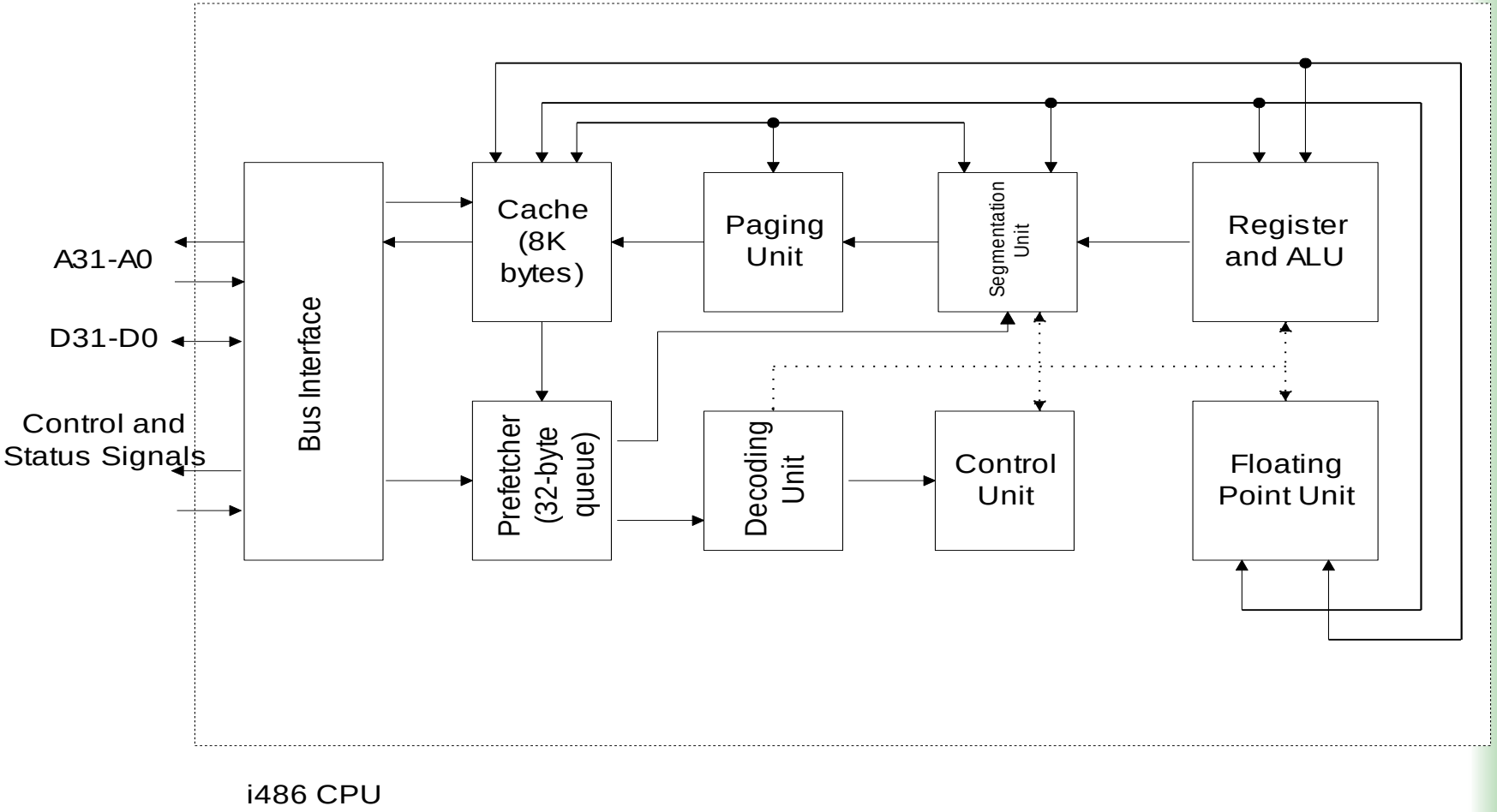
Segment Registers



80486: IA-32 with RISC elements

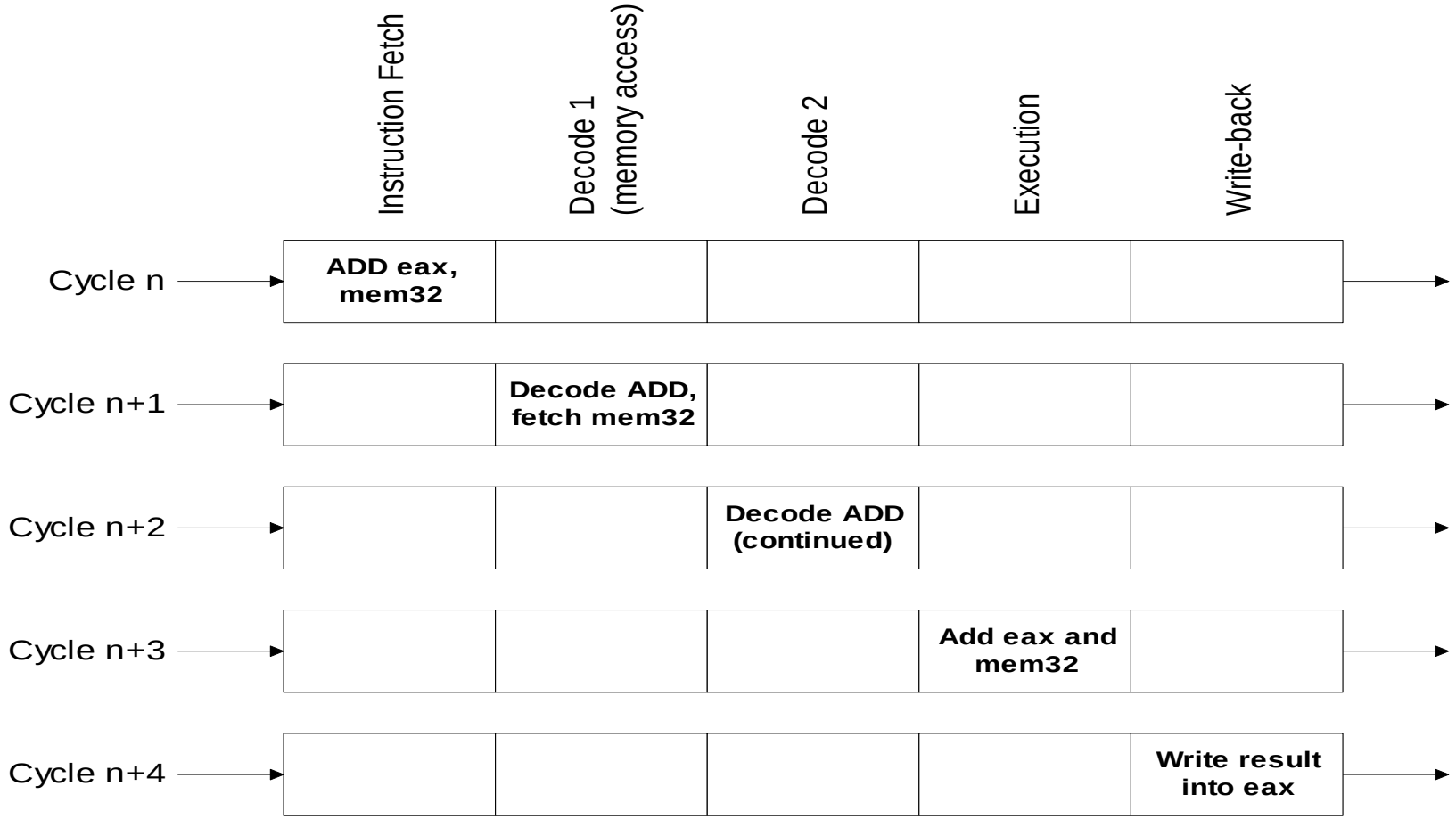
- Introduced 04/91
- Greatly improved 80386 CPU
- Hard-wired implementation of frequently used instructions (as in RISCs). On average 2 clock cycles/instruction.
- 5 stage instruction pipeline
- Internal L1 Cache Memory (8kB) + cache controller
- On-chip Floating Point coprocessor (FPU)
- Longer Prefetch Queue (32-bytes as opposed to 16 on the 80386)
- Higher frequency operation: up to 120MHz
- >1.2M transistors, 0.8μm CMOS. 168-pin PGA.

80486 Block Diagram



i486 CPU

80486 Pipeline



5th Gen. Processor: Pentium

- Pipelined FPU
  (2..10 times faster than 486 FPU. FDIV bug! Free replacement…)
  962,306,957,033 / 11,010,046 = 87,402.6282027341 (correct answer)
  962,306,957,033 / 11,010,046 = 87,399.5805831329 (flawed Pentium)
- Burst-mode bus cycles
  (fast data transfer from memory to cache)
- >3M transistors. BiCMOS. 0.8μm..0.35μm.
- Supply voltages: 5V..2.9V
- Packages: PGA273 and SPGA296
  (up to 16W power dissipation! Forced-convection cooling: fan)

5th Gen. Processor: Pentium

- Clock speeds: 60-266MHz
- Clock multiplier circuitry
  Processor runs faster than the system bus. Motherboard bus speeds 50, 60, 66MHz.
- System management mode (SMM)
  (full control over power management features)

5th Gen. Processor: Pentium

- Excellent Tutorial:
  http://developer.intel.com/software/products/itc/architec/ia32/pentdown.htm

Superscalar Pentium (1)

- Two almost independent integer pipelines and a floating point pipeline
- Branch Prediction
- Short command execution, through **many hardwired instructions**
- Binary compatibility for complex i386 instructions through a microprogrammed CISC unit
- Separate Code and Data caches, each with 8Kbytes, write-back strategy, conforming to the MESI cache coherency protocol

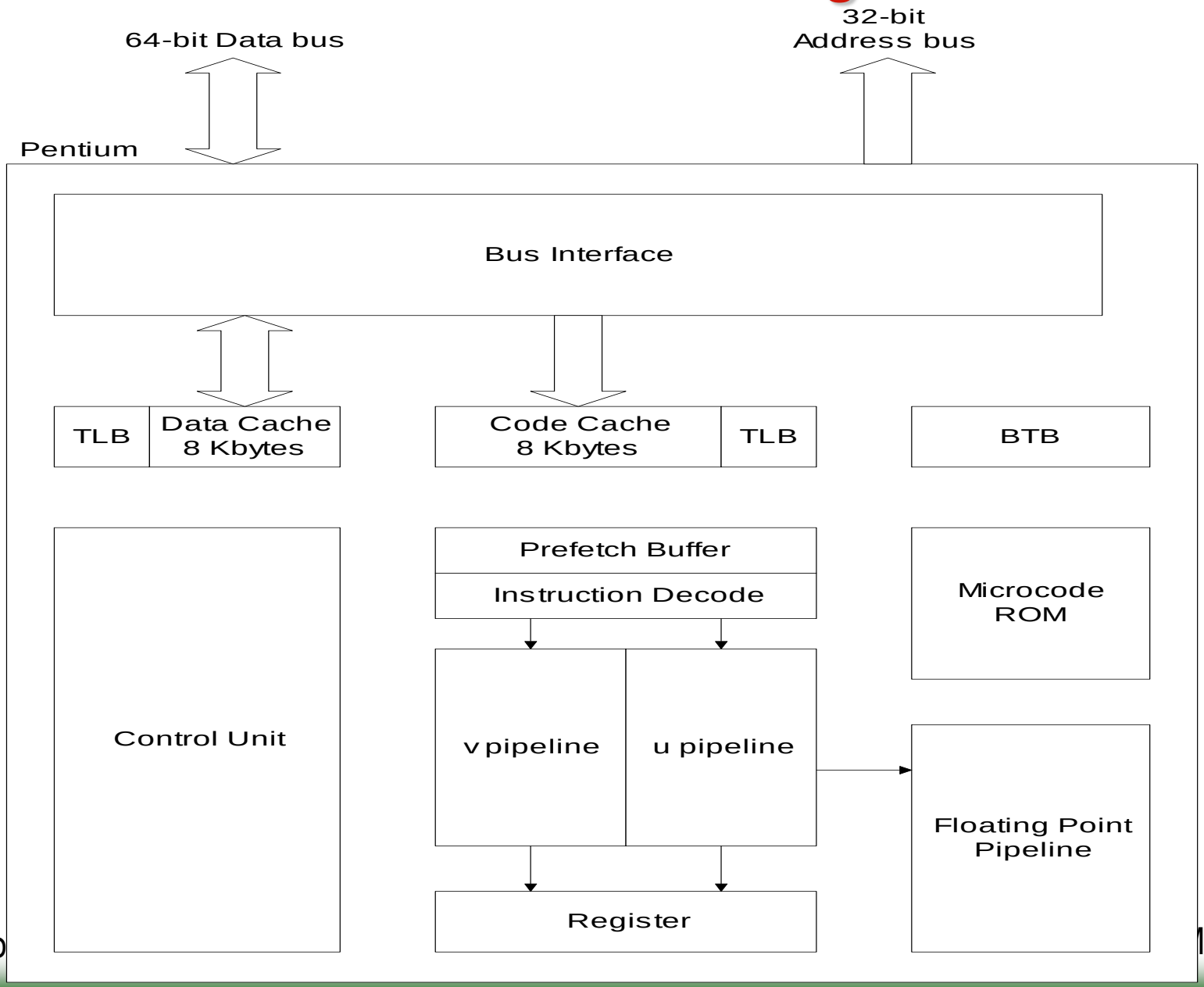
Superscalar Pentium (2)

- Wider 64-bit data bus, with burst mode for quicker cache line fills and write-backs
- Memory Management Unit for demand paging
- External bus speed of up to 66MHz
- Additional error detection functions such as internal parity check, self test and boundary scan test
- Execution tracing for external monitoring of instruction execution

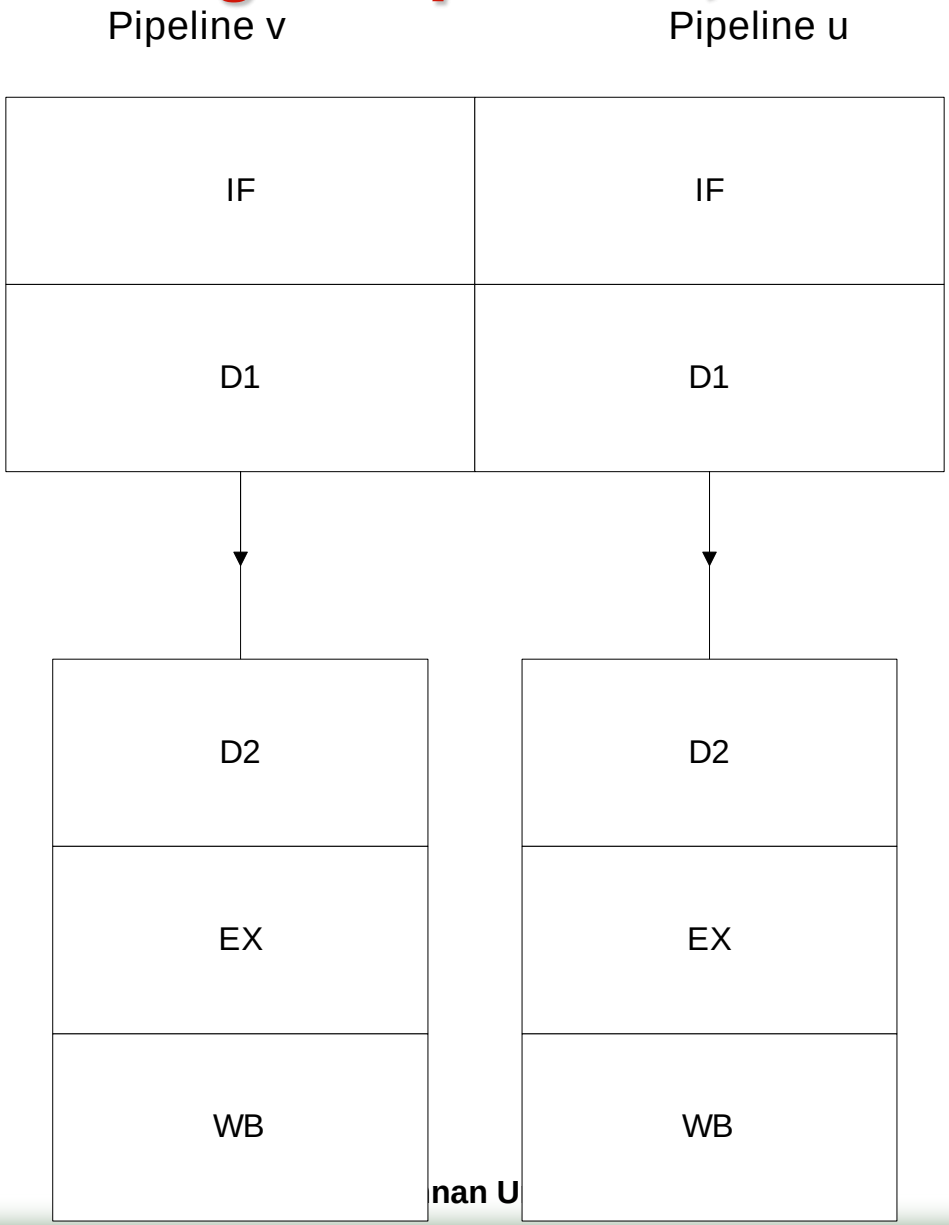
Superscalar Pentium (3)

- Real, virtual and protected 8086 modes
- Hardware debug through probe mode
- System management for implementing power save functions
- Performance monitoring to optimise code sequences
- Full binary compatibility with **all** x86 and x87 predecessors
- Dual Processor support with local on-chip APIC (Advanced Programmable Interrupt Controller)

Pentium Block Diagram



Integer Pipelines, u and v



Integer Pipelines, u and v

- Superscalar processors use more than one pipeline
- Under best case conditions the Pentium can complete *two* instructions in every clock
- IA-32 instructions have to be ‘paired’ according to Intel rules
  - Pipeline u can execute any IA-32 instruction
  - Pipeline v can execute ‘simple’ instructions
  - Pipeline u gets filled first
  - If the second instruction is NOT part of a pair – it waits for the next slot
- All pairing & decoding decisions are done in hardware –software support not required – but helps performance

Pipelines u and v, best case

Pipeline v - empty slot

Pipeline Operation

- Microcode unit can use BOTH u and v pipelines: Pentium microcode much quicker than i486
- Instruction fetch uses both prefetch buffers along with BTB (Branch Target Buffer) logic
- D1, D2 – decoding units: also implement pairing rules
- EX – execution stage
- WB – instructions can modify processor state and results can be written

Instruction Fetch & Branch Prediction

- Two 32-byte prefetch buffers operate along with the *Branch Target Buffer,* BTB
- When a branch instruction is fetched, the BTB predicts whether the branch will be taken or not
  - If Prediction == NOT TAKEN – continue linearly
  - If Prediction == TAKEN – enable other prefetch buffer
- If prediction wrong: flush pipelines and start again
- BTB uses past history to make predictions
- Correct predictions reduce pipeline stalls

Branch Prediction Example

```
for (k=i+prime; k<=SIZE; k+=prime)
     flags[k]=FALSE;
```

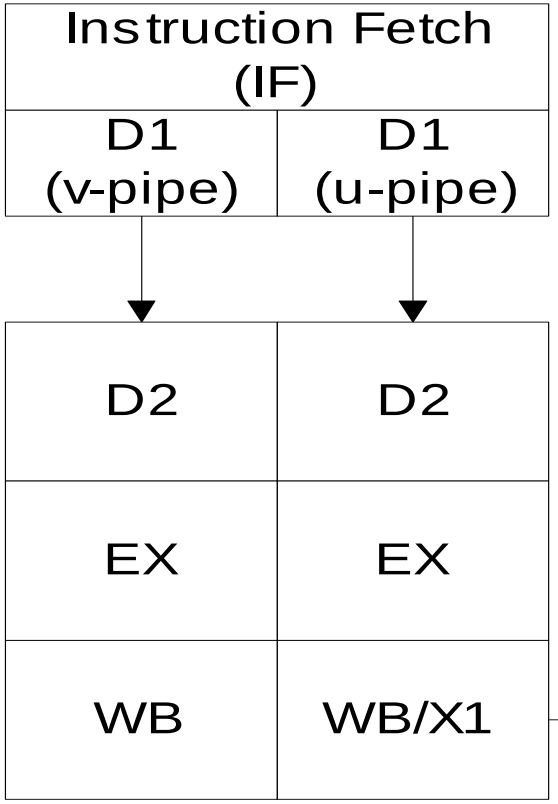
Becomes

```
inner_loop: mov   byte ptr flags[edx], al
                  add   edx, ecx
                  cmp   edx, SIZE
                  jle   inner_loop
```

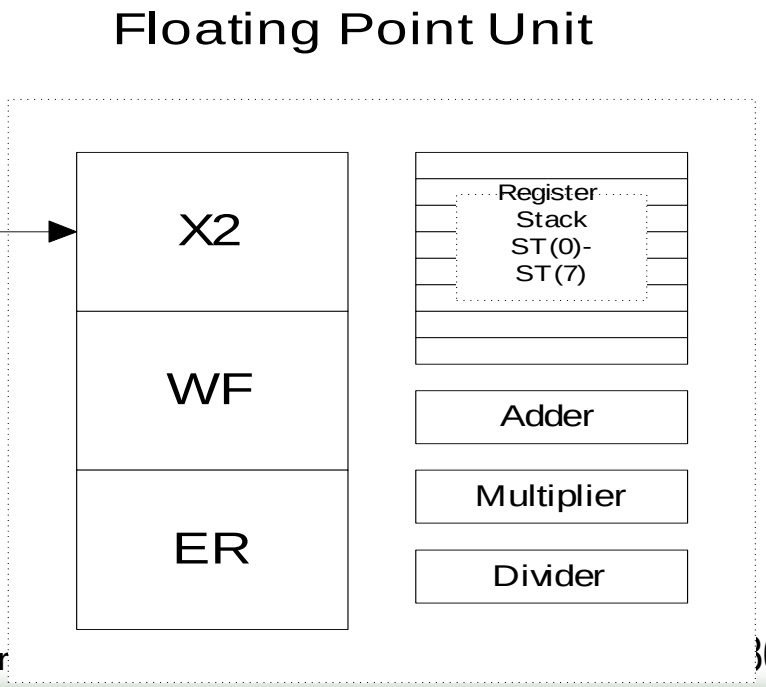
80486: Each iteration takes 6 clocks (branch taken costs 3 clock cycles)

Pentium: mov paired with add, cmp paired with jle. If prediction is for loop back, each iteration takes 2 clocks

Pentium Floating Point Pipeline



FP Pipeline has 8 stage
Shares first 4 stages with u integer pipeline
WB of U is first execution stage of FP pipline



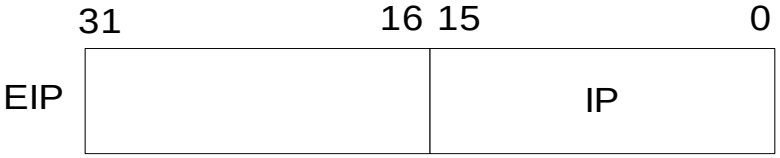
Cannot pair FP instructions (except FXCH) in the v pipeline

Pentium Registers

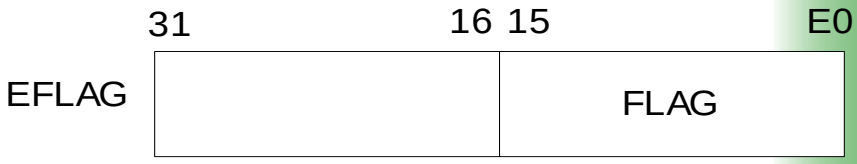
- Same register set as i386 – new flags for virtual 8086 and CPUID
- New control & test registers
  - CR4: debug support and 4M paging
  - TR12: selective activation of BTB etc

Pentium Registers(1)

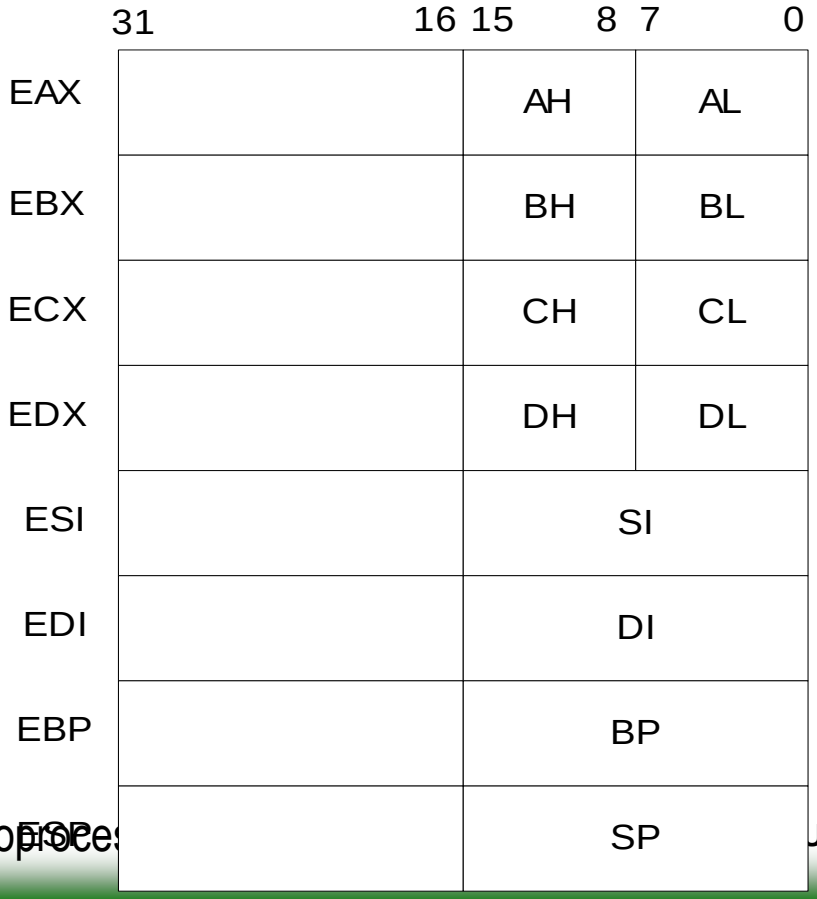
Instruction Pointer



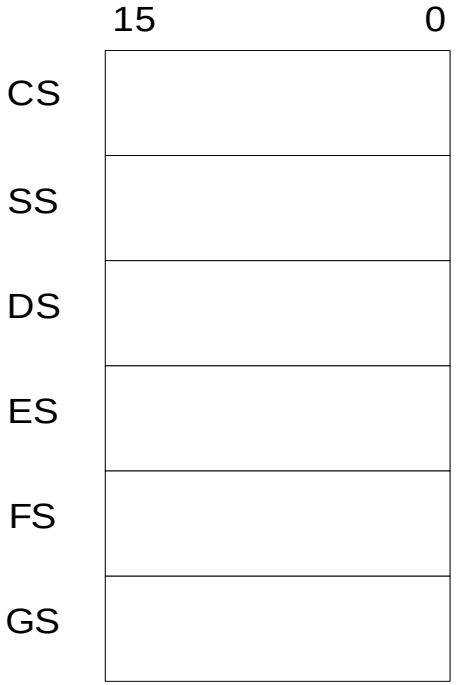
EFLAG Register



General-Purpose Registers



Segment Registers



i386 Control Registers

	15 0	31 0	19 0
TR	TSS Selector	TSS Base Address	TSS Limit
LDTR	LDTSS Selector	LDT Base Address	LDT Limit
IDTR		IDT Base Address	IDT Limit
GDTR		GDT Base Address	GDT Limit

Control Registers

	31 16	15 0
CR3		
CR2		
CR1		
CR0		

Test Registers

	31 16	15 0
TR7		
TR6		

Debug Registers

	31 16	15 0
DR7		
DR6		
DR5		
DR4		
DR3		
DR2		
DR1		
DR0		

Pentium Registers(2)

	15 0	31 0	19 0
TR	TSS Selector	TSS Base Address	TSS Limit
LDTR	LDTSS Selector	LDT Base Address	LDT Limit
IDTR		IDT Base Address	IDT Limit
GDTR		GDT Base Address	GDT Limit

Control Registers

	31 16	15 0
CR4		
CR3		
CR2		
CR1		
CR0		

Test Registers

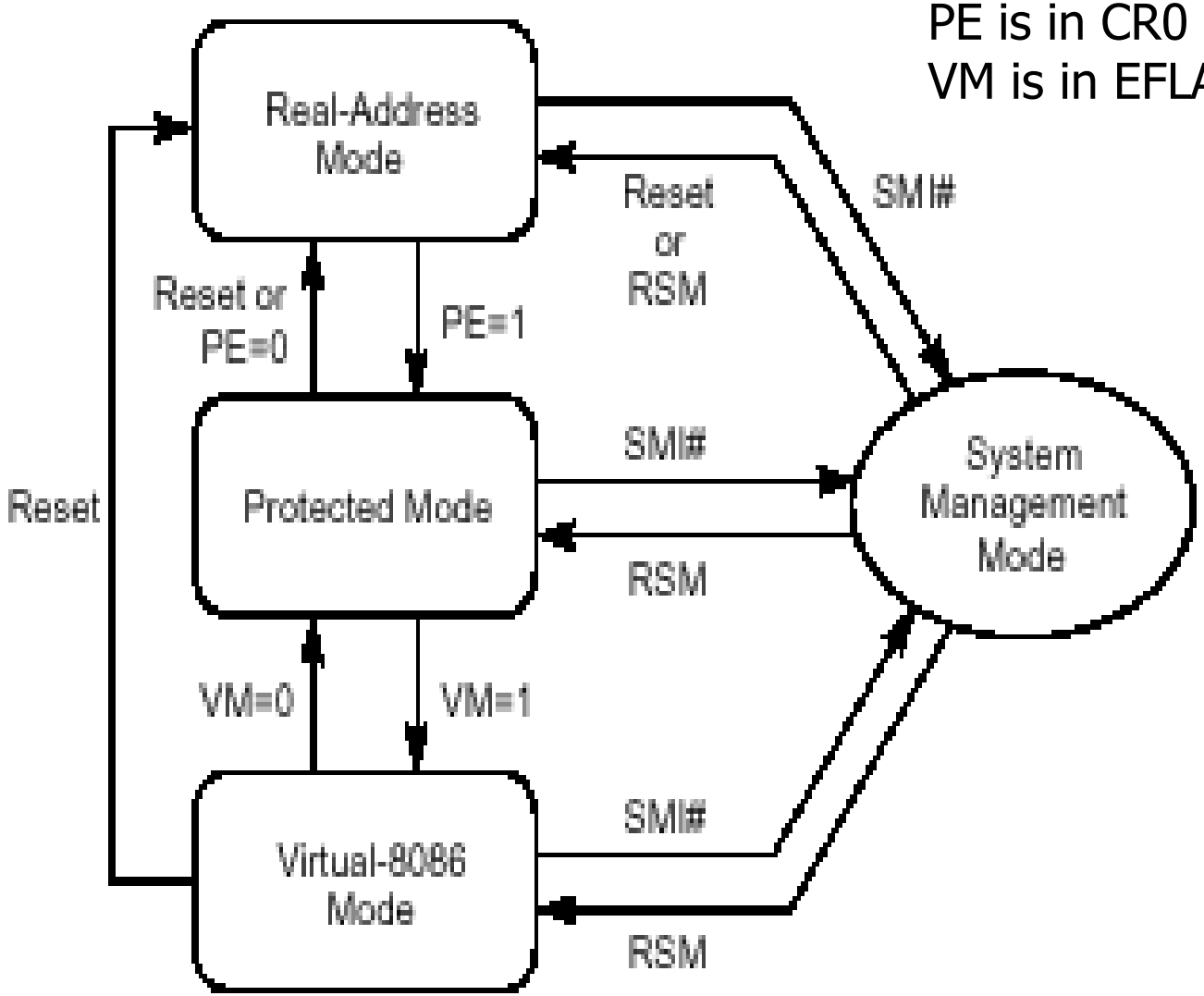
	31 16	15 0
TR12		
TR7		
TR6		

Debug Registers

	31 16	15 0
DR7		
DR6		
DR5		
DR4		
DR3		
DR2		
DR1		
DR0		

Pentium Operating Modes

PE is in CR0
VM is in EFLAGS



Remember: Real Mode Segmentation

- Real Mode (8086): 20 address bits
- Use 2 x 16 bit regs to make up 20 bits between them
- Segment Regs (CS, DS, SS etc) + Offset Registers (IP, SP, BX, SI etc)

Abbreviations:

RM: Real Mode
PM: Protected Mode

RM: 8086 compatibility

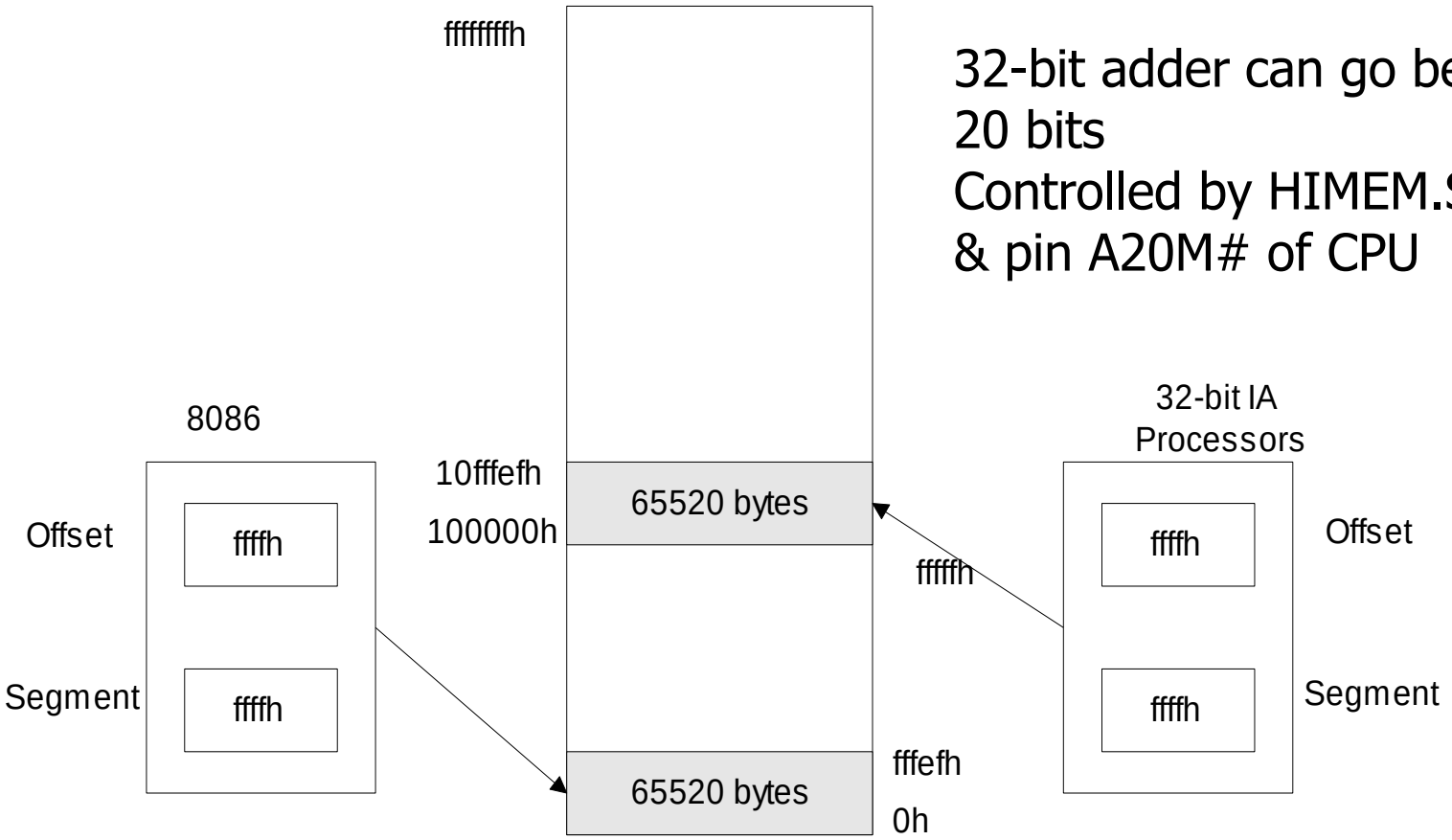
Compatibility with exceptions…
Example: Segment register: 0FFFFh. Offset: 0FFFFh

8086	
Segment *16	0FFFF0h
+ Offset	0FFFFh
Linear Address	0FFEFh

8086: 20-bit adder
Pentium: 32-bit adder

Pentium	
Segment *16	0FFFF0h
+ Offset	0FFFFh
Linear Address	10FFEFh

RM: 8086 compatibility



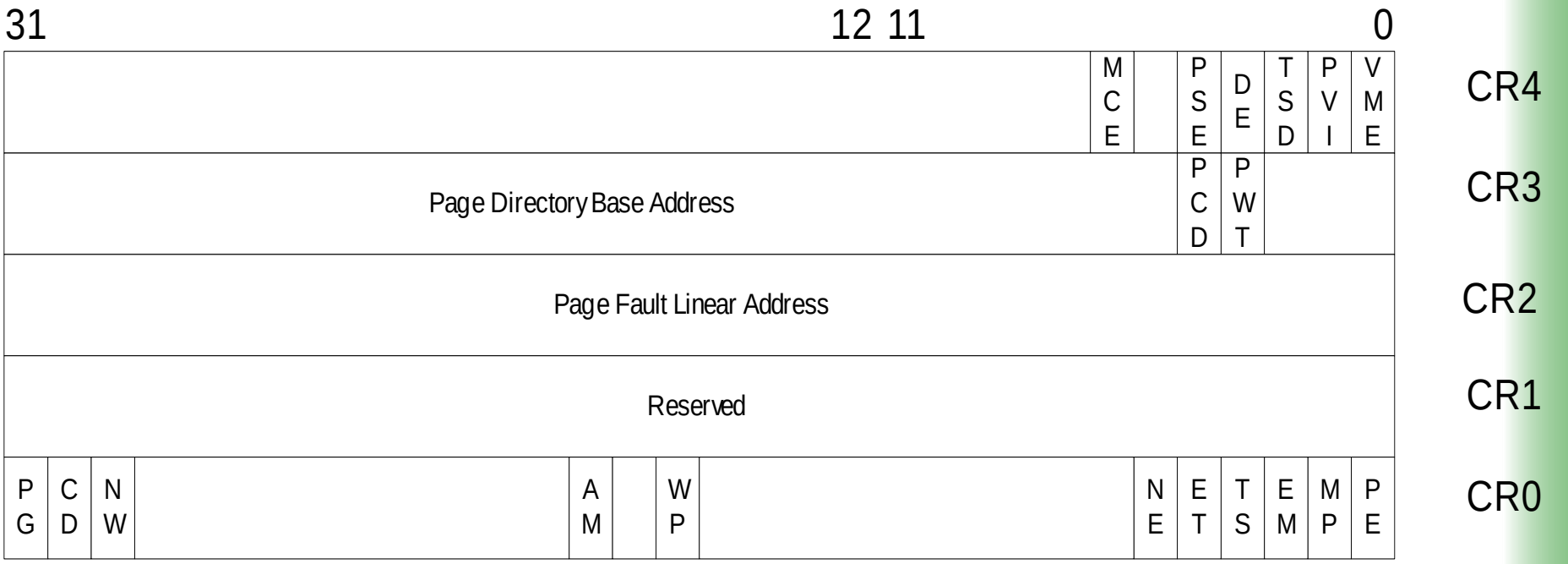
PM Memory Management

- PM Memory Management through
  - (Improved) Segmentation
  - Paging
- Use improved segmentation and paging to implement a memory protection scheme
- Protected Mode: Offset registers like EIP, ESP etc are all 32-bits

Segmentation and Paging

- Segmentation
  - Protects tasks from interfering with each other (e.g. prevents them from writing into each others memory areas)
  - Assigns priority levels to tasks
- Paging
  - Supports Demand-Paged Virtual Memory
  - Paging can also be used for protection and permissions

Pentium CR0-CR4 Control Registers



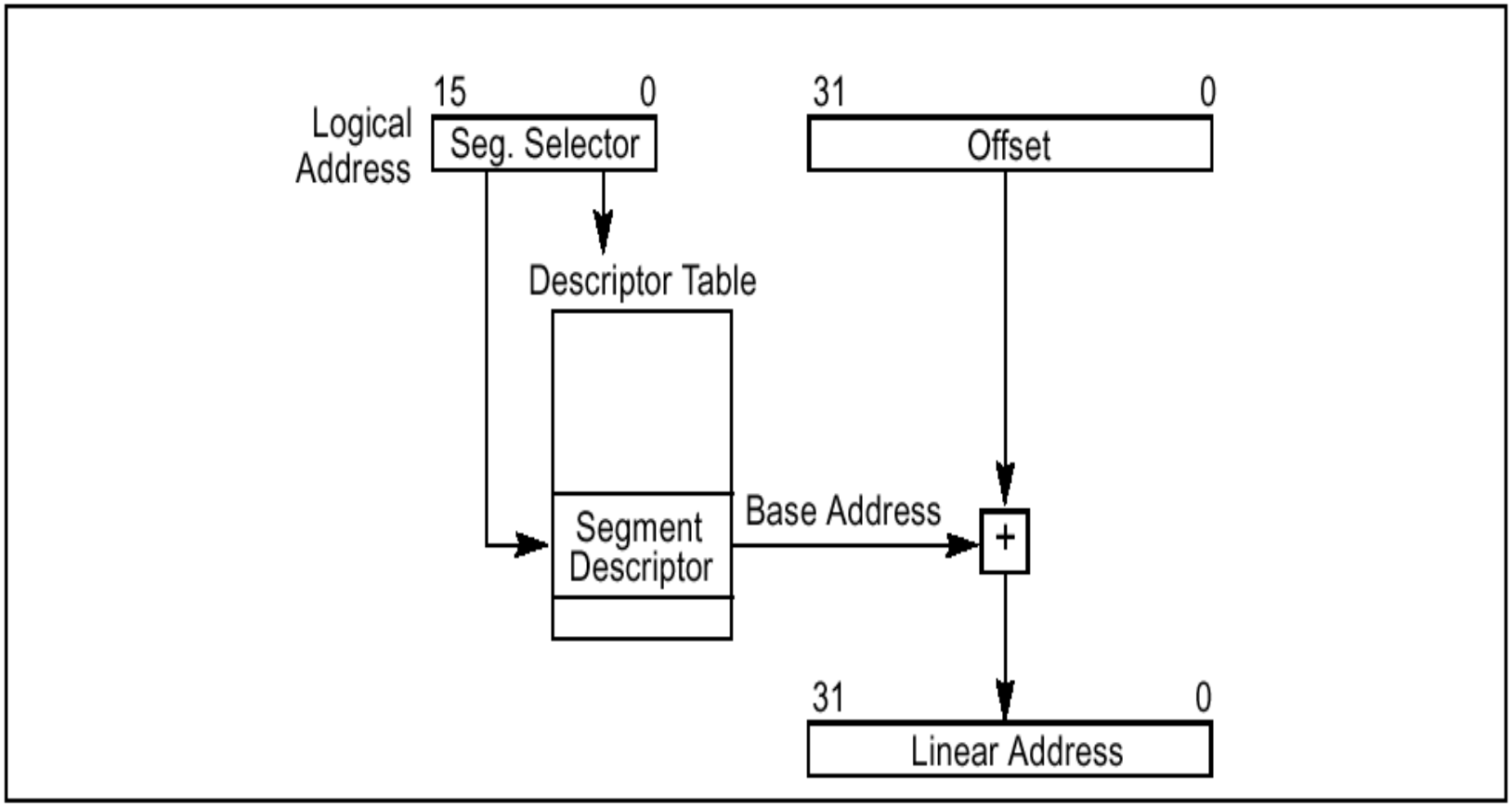
Bit 0 of CR0: Enables Protected Mode

Bit 31 of CR0: Enables Paging

Segmentation in PM

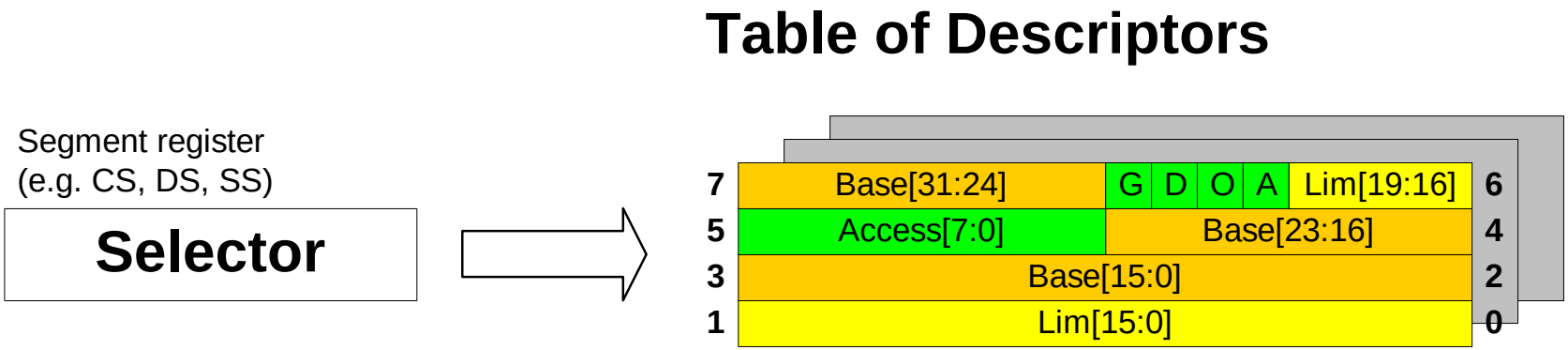
- Segment Registers have an *entirely* different meaning in PM
  - Content of Segment Register in RM: Segment Address
  - Content of Segment Register in PM: Segment Selector
- Segment Selectors are pointers into a Table of Segment Descriptors (Descriptor Table)
- Segment descriptors fully specify segments (Base address, size, access rights)

Logical and Linear Addresses



Segment Selector and Descriptor

The Segment Selector points into a Table of Segment Descriptors.



- Descriptor describes segment:
  - Base Address. Size (Limit). Access rights.

Segment Selector

Segment Register Format (PM)

- Selector[12:0]
  - Selects entry in descriptor table
- Table Indicator TI
  - 0: Global Descriptor Table. 1: Local Descriptor Table
- Requested Priority Level RPL[1:0]
  - 00b: Highest. 11b: Lowest

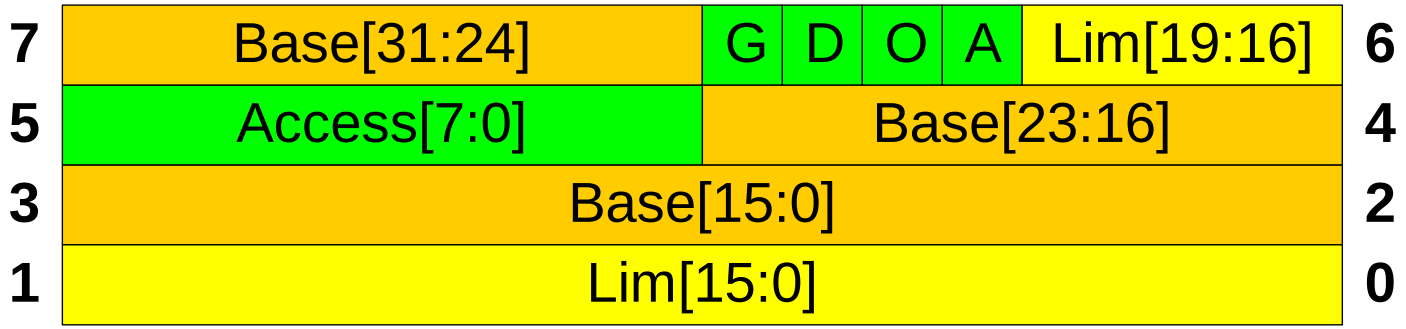
Q: How many segment descriptors can a descriptor table contain?

Segment Selector

- Selector selects one of 8192 descriptors from one of two tables of descriptors
- Descriptor describes Segment
  - Location (Base address)
  - Length (Limit)
  - Access Rights
- Descriptor tables are located in memory

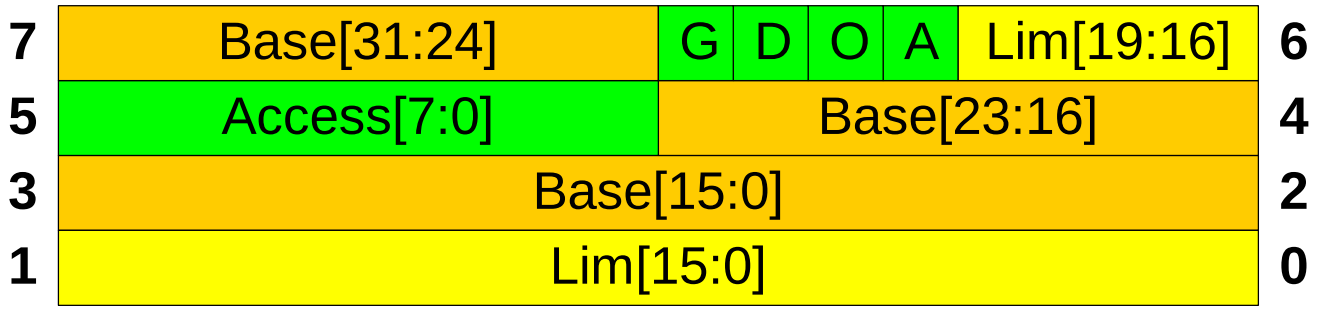
Segment Selector

- Each descriptor is 8 bytes in lengths



- 32 bits for base address (anywhere in 4GB address space)
- 20 bits for segment size
  Segment size can be 1B-1MB, or 4kB-4GB (in 4KB steps) depending on granularity bit G
- Limit contains the last valid offset address of the segment

Segment Protection



- Any attempt to
  - access a segment beyond its limits or
  - access a segment without sufficient access rights
- results in a **GENERAL PROTECTION FAULT** interrupt

Descriptor Tables

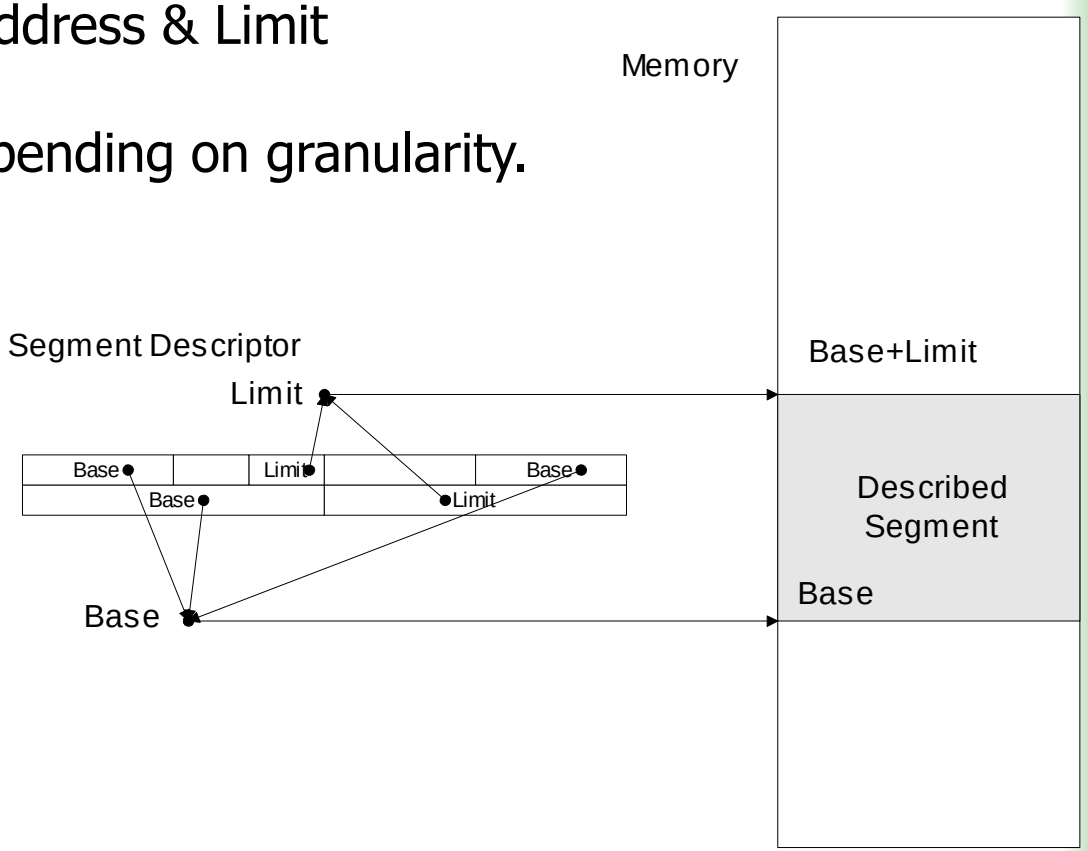
- There are two descriptor tables:
  - Global Descriptor Table (GDT)
    - Contains segment descriptors that apply to all applications
  - Local Descriptor Table (LDT)
    - Unique to an application
    - LDT changes when task changes

Descriptor Tables

- Q: How many descriptors are available to an application at any time?
- Q: How much memory is needed for a fully occupied descriptor table?

Segment Descriptor

Segment Descriptor is kept in GDT or LDT
Descriptor contains Base Address & Limit
+ Access Types
Maximum segment size depending on granularity.
Byte Gran: Max Limit=1MB
4K Gran: Max Limit=4GB



Descriptor Table Registers

	15 0	31 0	19 0
TR	TSS Selector	TSS Base Address	TSS Limit
LDTR	LDTSS Selector	LDT Base Address	LDT Limit
	IDTR	IDT Base Address	IDT Limit
	GDTR	GDT Base Address	GDT Limit

PM Segmentation Exercise

Segment Register Format (PM)

Selector [12:0]	TI	RPL[1:0]

Descriptor Format

7	Base[31:24]	G	D	O	A	Lim[19:16]	6
5	Access[7:0]	Base[23:16]					4
3	Base[15:0]						2
1	Lim[15:0]						0

BX

00h	02h

DS

00h	10h

GDT

Descriptor 2	00h	00h
	92h	10h
	00h	04h
	00h	0Bh
Descriptor 1	00h	00h
	92h	10h
	00h	00h
	00h	03h
Descriptor 0	00h	00h
	92h	10h
	00h	10h
	00h	07h

GDT Base Address

Memory System

01h	0CAh	*Adr 100016h*
22h	10h	*Adr 100014h*
10h	02h	*Adr 100012h*
0FFh	0E0h	*Adr 100010h*
0F1h	00h	*Adr 10000Eh*
92h	8Fh	*Adr 10000Ch*
1Eh	20h	*Adr 10000Ah*
21h	47h	*Adr 100008h*
43h	30h	*Adr 100006h*
0A5h	41h	*Adr 100004h*
13h	17h	*Adr 100002h*
55h	0AAh	*Adr 100000h*

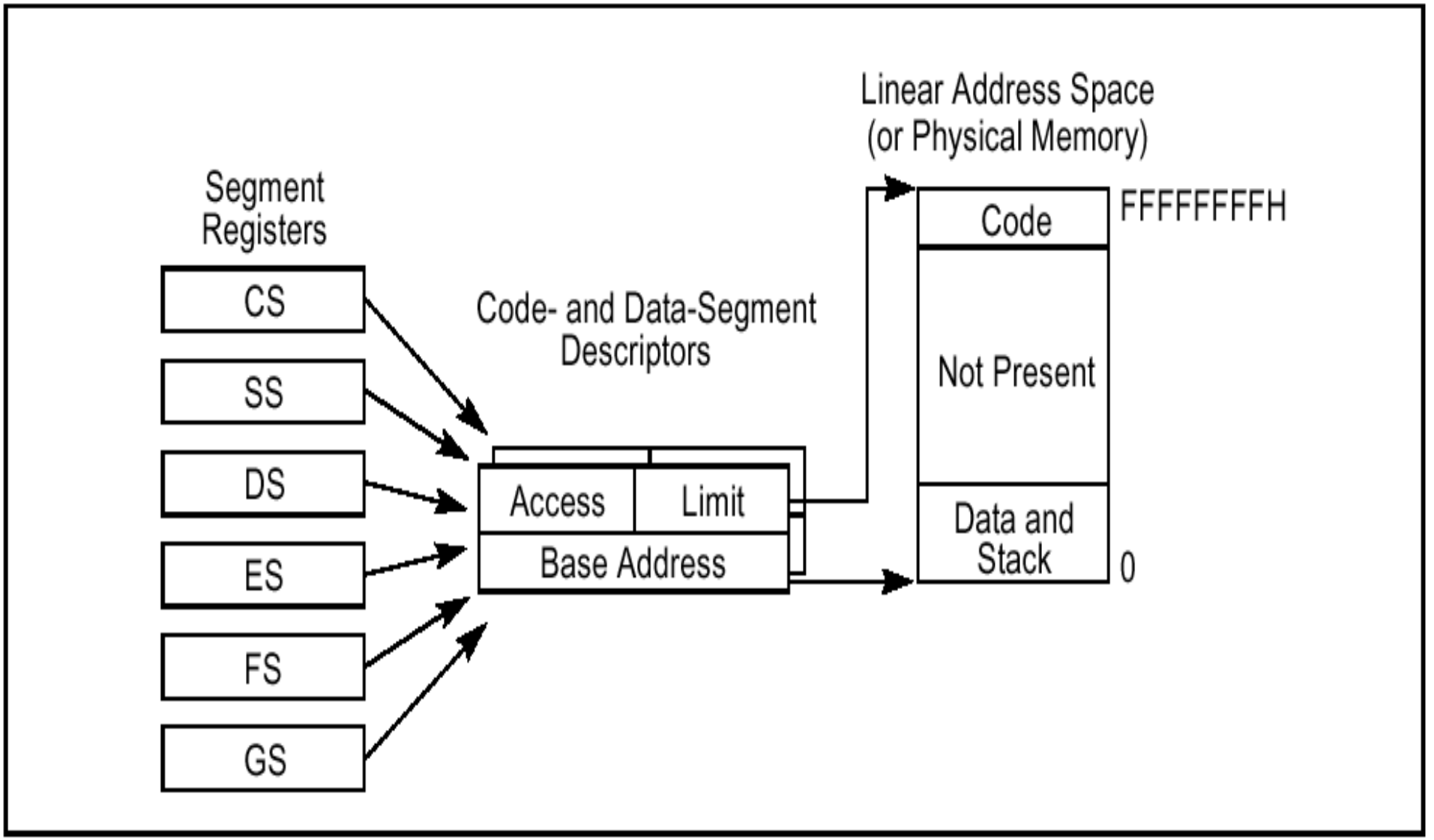
Segmentation Example

- Consider instruction MOV AL, DS:[BX]
  - Determine the offset within the selected segment
  - Which descriptor is selected?
  - Determine the base addresses and sizes of all segments described in the descriptor table
  - Which data byte is moved into register AL?
  - What happens if BX is set to 000Ch before MOV AL, DS:[BX] is executed again?

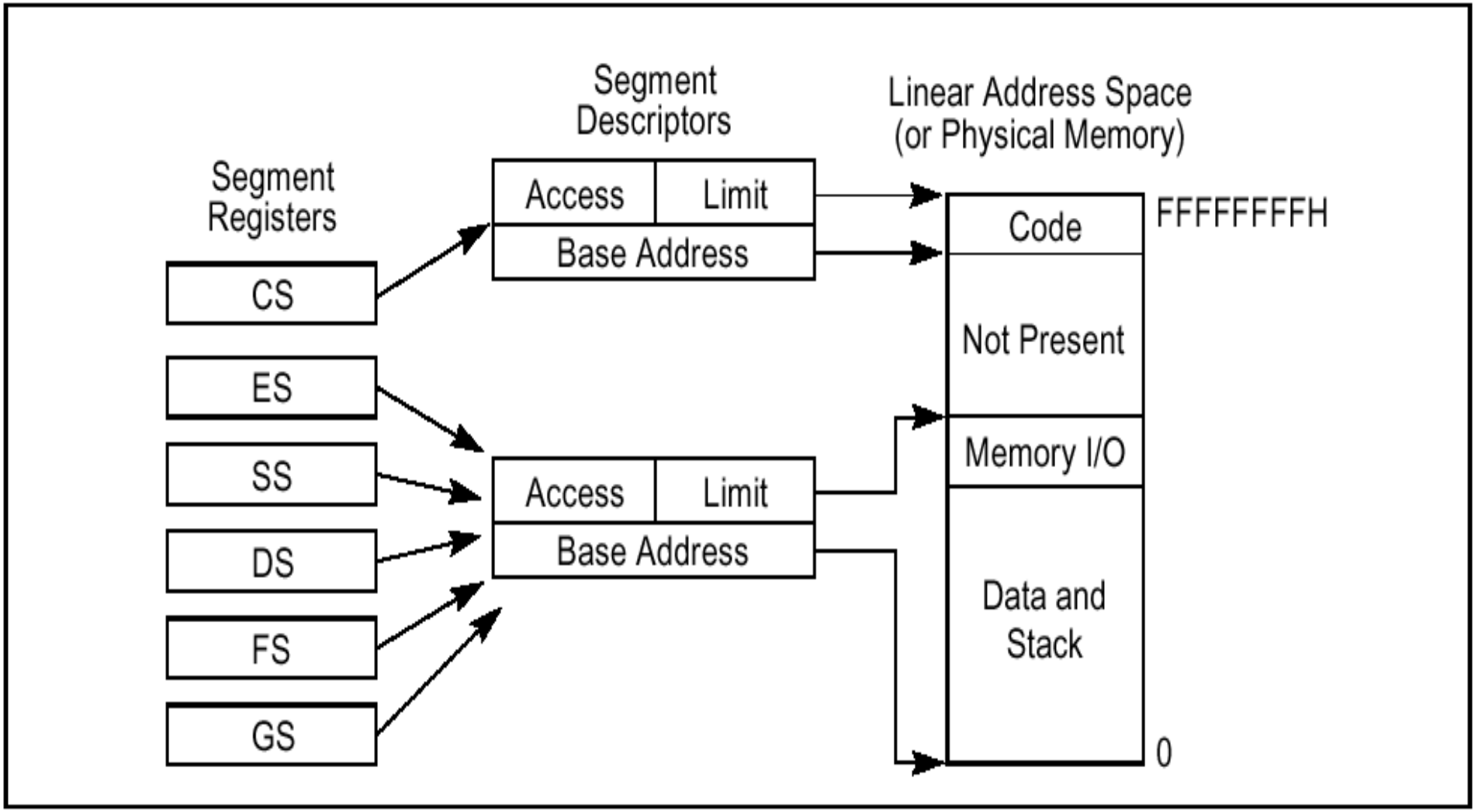
Segmentation Schemes

- Basic Flat Segmentation
  - Not really segmented at all
- Protected Flat Segmentation
  - As used in Windows NT
  - Only slight improvement on Basic
- Multisegment Model
  - All segments are unique

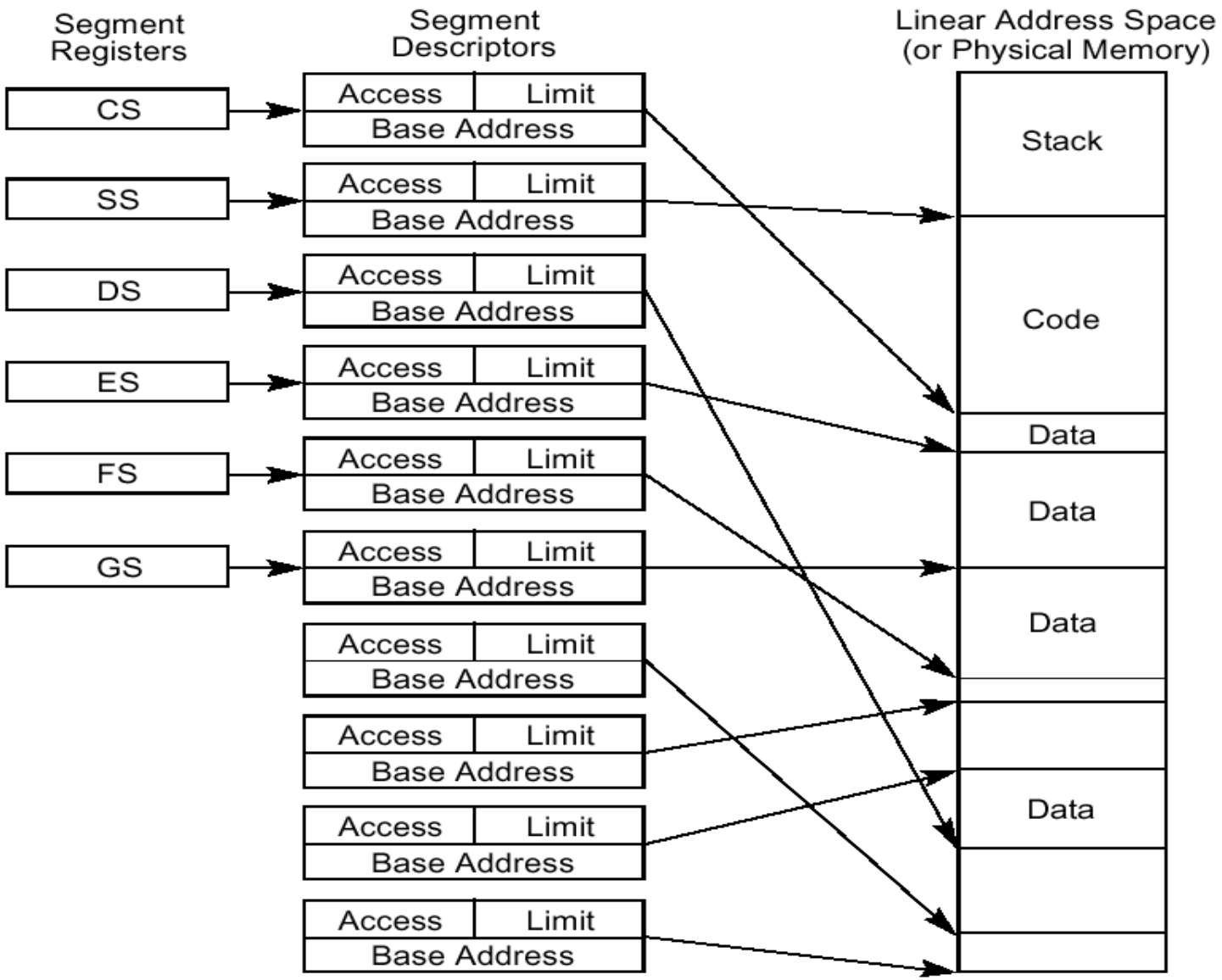
Basic Flat Segmentation



Protected Flat Model



Multisegment Model



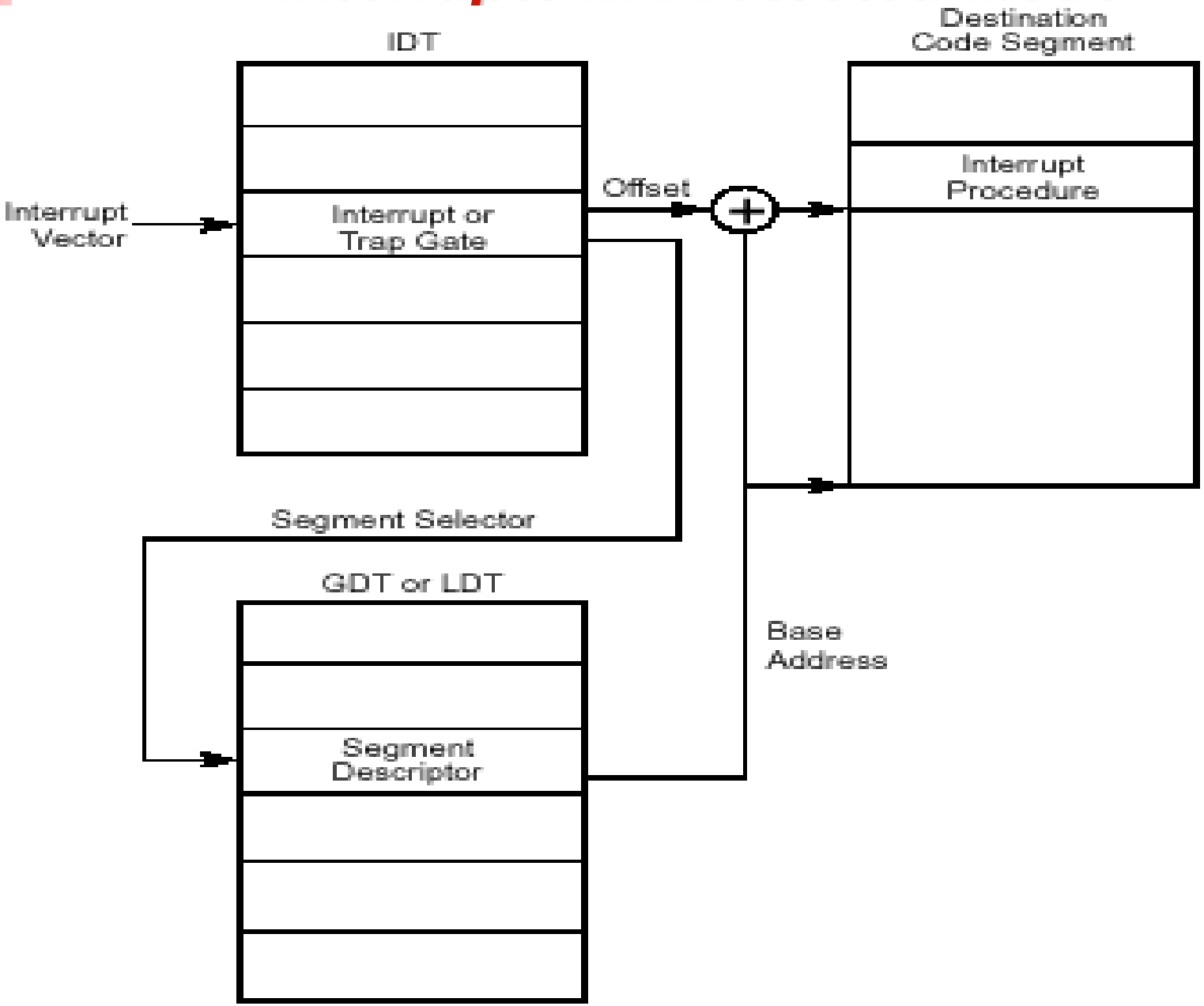
Segmentation: Effects on Performance

- Descriptor tables stored in memory
- Everytime the processor accesses memory, it has to read an element from a Descriptor Table itself in memory before completing the memory access
- Overhead: instead of one memory access we now have two
- Solution: keep a COPY of the current table entries in the CPU itself (descriptor cache)
- These are called HIDDEN REGISTERS

Hidden Registers

To cut down memory accesses to tables the processor loads data into hidden registers associated with each segment register
These copies are used rather than external memory versions to speed up access

Interrupts in Protected Mode



Demand-Paged Virtual Memory

- Demand-Paged Virtual Memory
- Uses a combination of main memory (semiconductor memory, RAM) and system hard-disk to make it appear to applications as if the computer has access to a much larger main memory
- Allows swapping of pages in and out of memory

Paging in IA-32 Architecture

- 16-bit segment registers combined with 32-bit offset registers allow an address space of 64 TB (1TB=2^{40})
- But i386 - Pentium can only address 4 GB using its address bus
- What's more, the available memory in a PC is (still) less than 4 GB
- Q: How much RAM would you find in today's PCs?

Paging in IA-32 Architecture

- Paging maps a large linear address space onto the smaller address space of main memory plus the large address space of the hard disk
- Page sizes
  - 386/486: 4kB
  - Pentium: 4kB or 4MB

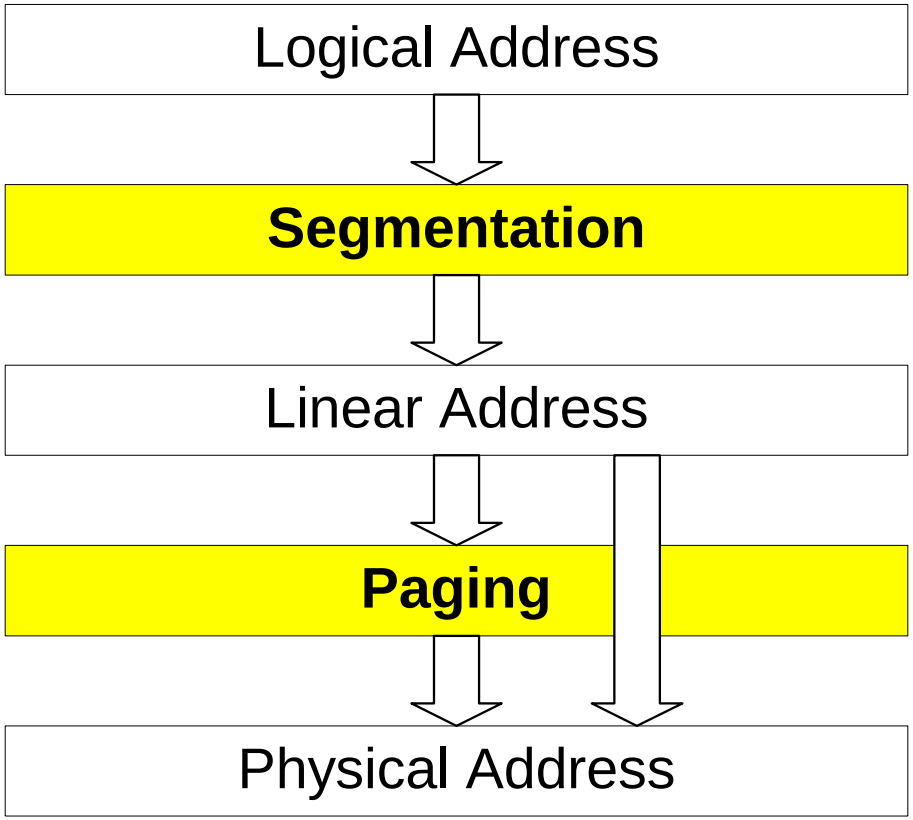
Paging in IA-32 Architecture

- If a page is not currently in memory and some code tries to access it…
  a page exception occurs
- The operating system has to serve the paging exception and swap pages

IA-32 Paging

- If enabled, the Paging System works on the 32-bit Linear Address produced by the Segmentation System
- Windows NT uses Protection mechanisms in the IA-32 Paging rather than Segmentation to protect Tasks and implement priorities

IA-32 Paging



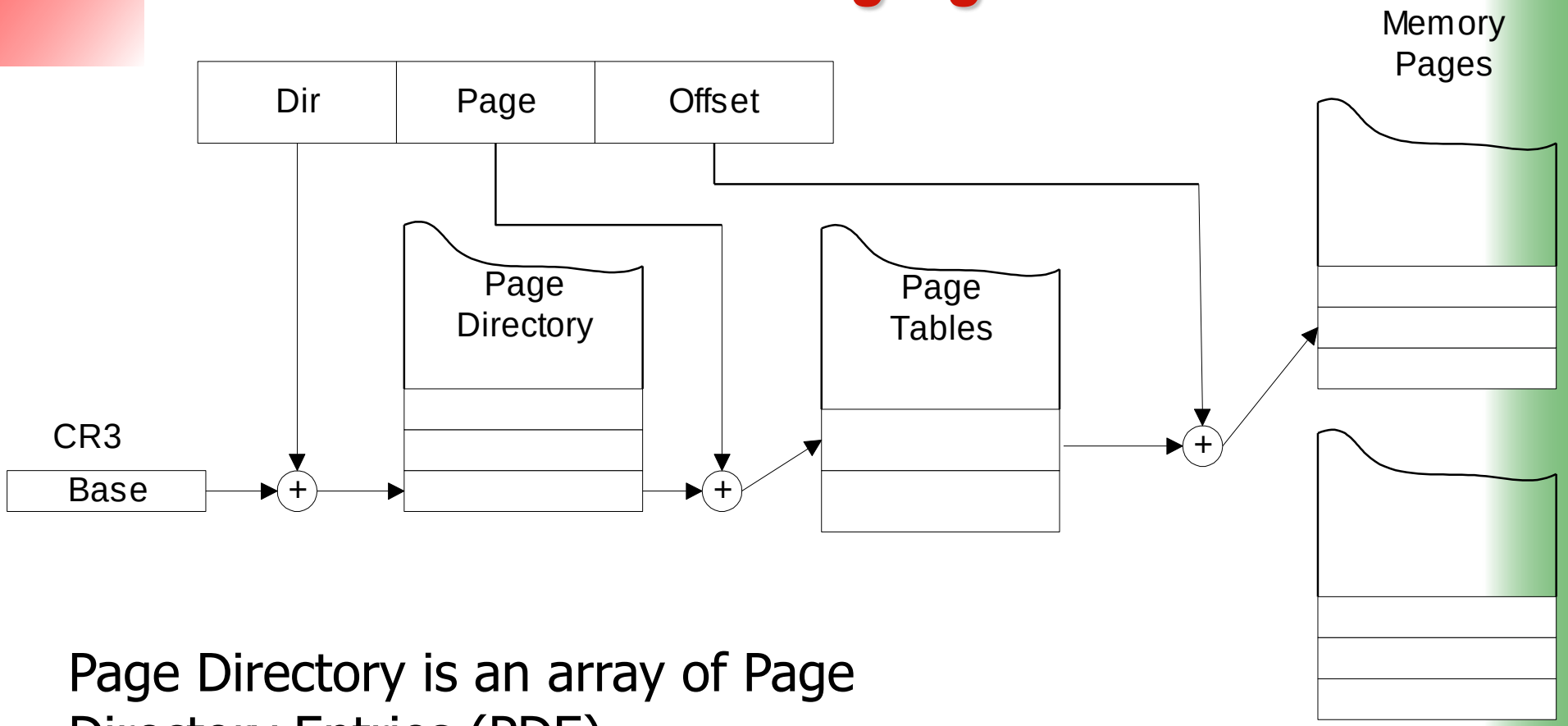
With the memory paging unit, the linear address is invisibly translated into any physical address.

IA-32 Paging

Linear Address

- Linear address is broken into 3 sections:
  - Page Directory Entry (PDE) - 10 bits
  - Page Table Entry (PTE) - 10 bits
  - Page offset address - 12 bits

4K Paging



Page Directory is an array of Page Directory Entries (PDE)

Page Table is an array of Page Table Entries (PTE)

User/Supervisor field in PDE & PTE allows NT to implement protection of pages

IA-32 Paging

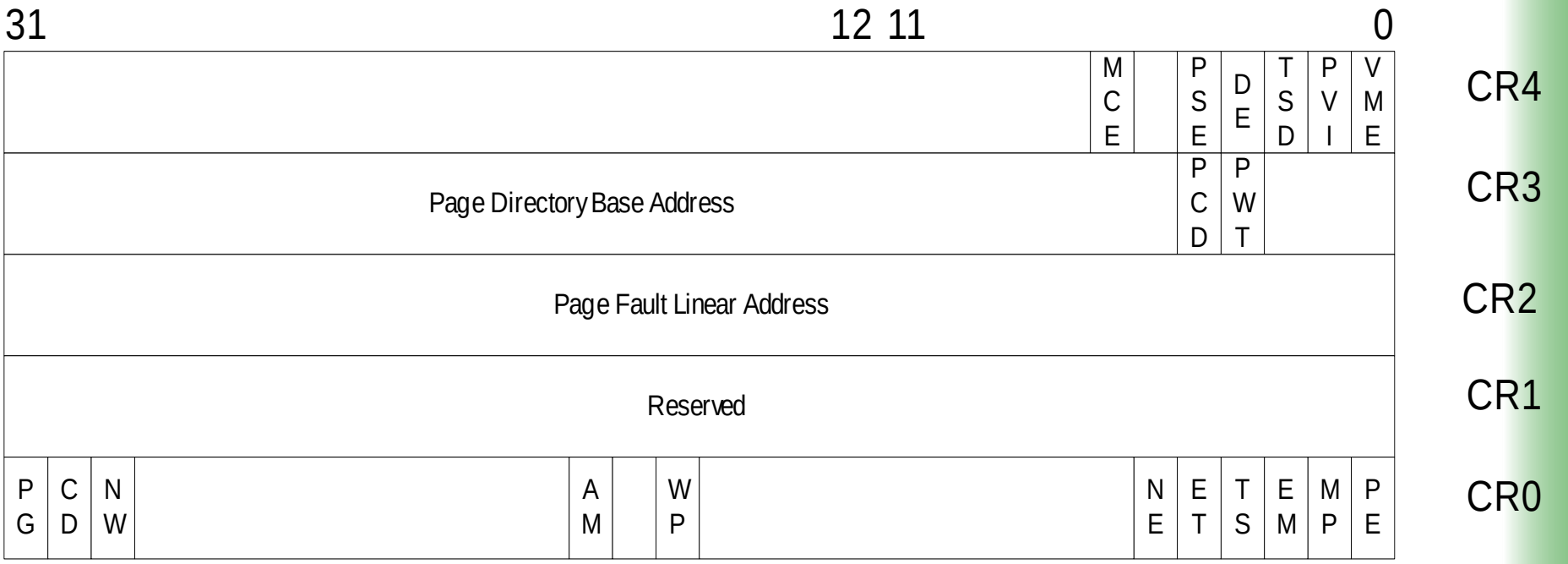
Linear Address

- Page Directory contains up to 1024 doubleword (32b) entries that point to 1024 page tables.
- Page Table contains up to 1024 doubleword (32b) entries

IA-32 Paging

- Q: If the entire 4GB of memory are paged, how much memory has to be allocated for the page tables, and the page directory?

Pentium CR0-CR4 Control Registers

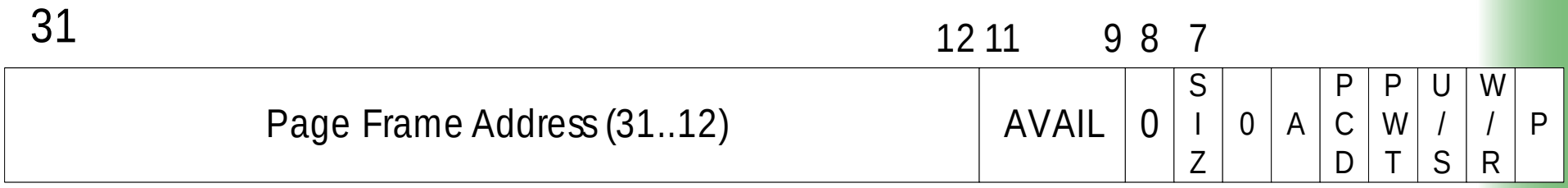


Bit 31 of CR0 turns Paging On (1) or off (0).

CR3: Page Directory Base Address and the memory interface bits PCD and PWT.

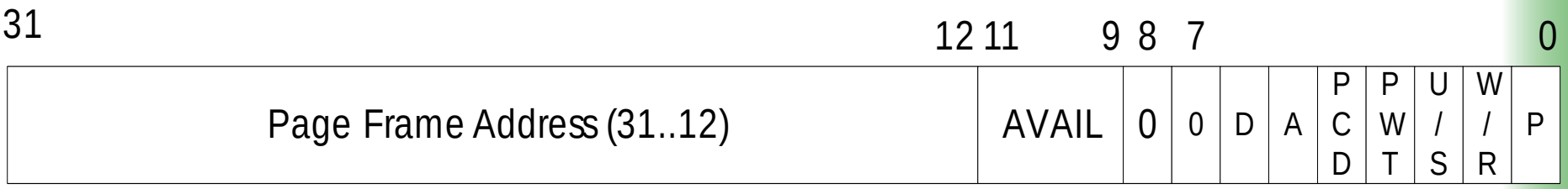
CR4: Controls 4M/4K paging

Page Directory Entry (4K)



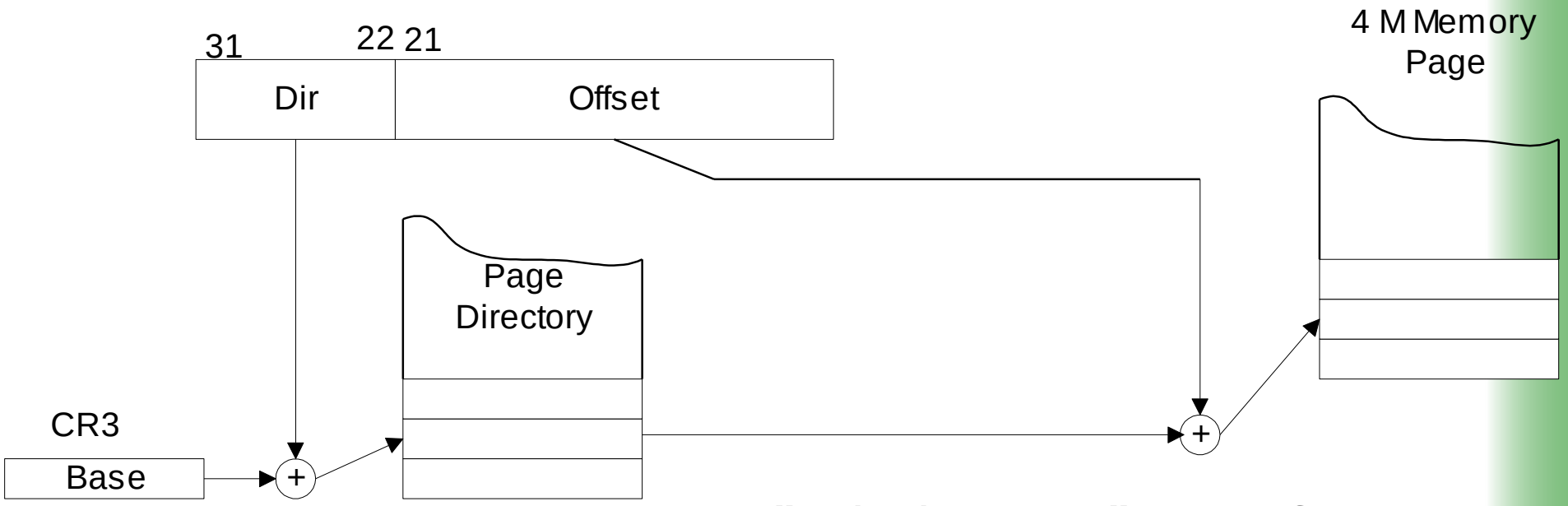
AVAIL	Available for OS	
SIZ (page size)	0: 4Kbyte	1: 4Mbyte
D (dirty)	0: page has not been overwritten	1: page has been overwritten
PCD (page cache disable)	0: page is cacheable	1: page NOT cacheable
PWT (page write-through)	0: page uses write-back strategy	1: page uses write-through strategy
A (accessed)	0: page has not yet been accessed	1: page has been accessed
U/S (page access level – User/Supervisor)	0: supervisor (CPL = 0..2)	1: user (CPL =3)
R/W (read/write)	0: page is read only	1: page is writeable
P (page present)	0: page is swapped to disk	1: page is resident in memory

Page Table Entry (4K)



AVAIL	Available for OS	
SIZ (page size)	0: 4Kbyte	1: 4Mbyte
D (dirty)	0: page has not been overwritten	1: page has been overwritten
PCD (page cache disable)	0: page is cacheable	1: page NOT cacheable
PWT (page write-through)	0: page uses write-back strategy	1: page uses write-through strategy
A (accessed)	0: page has not yet been accessed	1: page has been accessed
U/S (page access level – User/Supervisor)	0: supervisor (CPL = 0..2)	1: user (CPL =3)
R/W (read/write)	0: page is read only	1: page is writeable
P (page present)	0: page is swapped to disk	1: page is resident in memory

Pentium: 4M Paging



No "sub-directory" type of structure in 4Mbyte pages

Page Directory Entry (4M)

AVAIL	Available for OS	
SIZ (page size)	0: 4Kbyte	1: 4Mbyte
D (dirty)	0: page has not been overwritten	1: page has been overwritten
PCD (page cache disable)	0: page is cacheable	1: page NOT cacheable
PWT (page write-through)	0: page uses write-back strategy	1: page uses write-through strategy
A (accessed)	0: page has not yet been accessed	1: page has been accessed
U/S (page access level – User/Supervisor)	0: supervisor (CPL = 0..2)	1: user (CPL =3)
R/W (read/write)	0: page is read only	1: page is writeable
P (page present)	0: page is swapped to disk	1: page is resident in memory

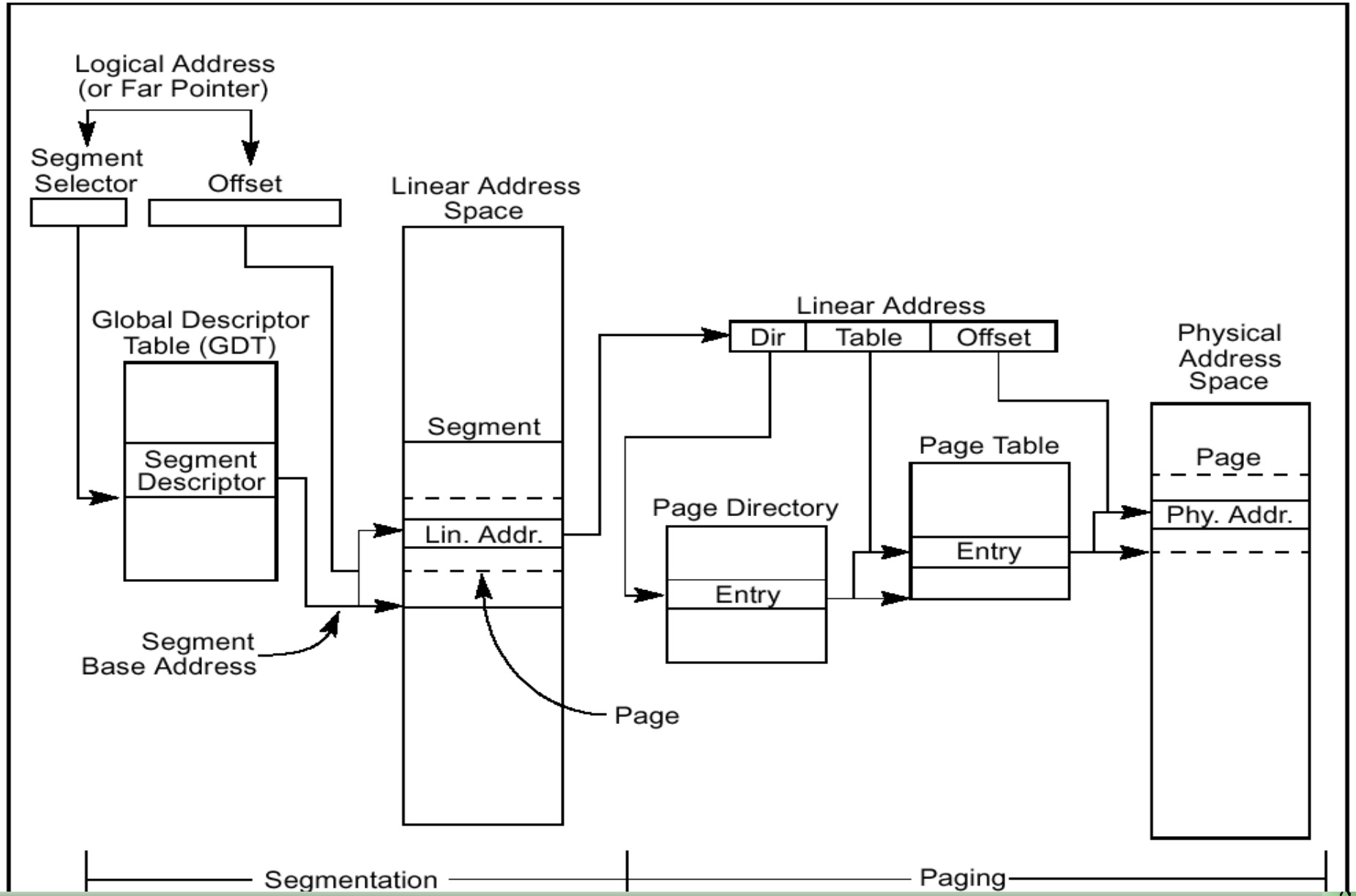
Paging: Effect on Performance

- For each memory access the processor needs to access the Page Directory and the Page Table to find the Page Frame Address to Add to the Page offset
- Potentially adverse effect on Performance
- Solution use a copy of recently used PDE & PTE elements
- Called Translation Lookaside Buffers (TLB)

Translation Lookaside Buffers (TLB)

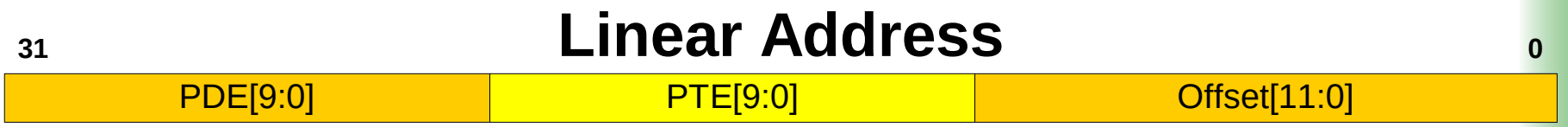
- TLB: a Cache of recently-used Page Translations
- If Page Table Entry is in the TLB Pentium uses TLB access to avoid a memory accesses
- If Entry Not in TLB – 2 access required – and new table placed in TLB
- Pentium: Each of the Code and Data Caches has its own TLB

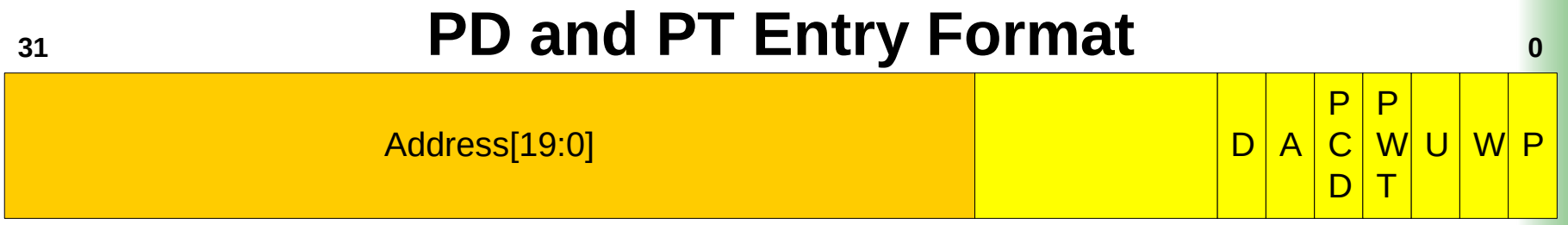
Segmentation and Paging



Paging Exercise

- Remember the sections in a linear address, and the PD and PT entry formats…

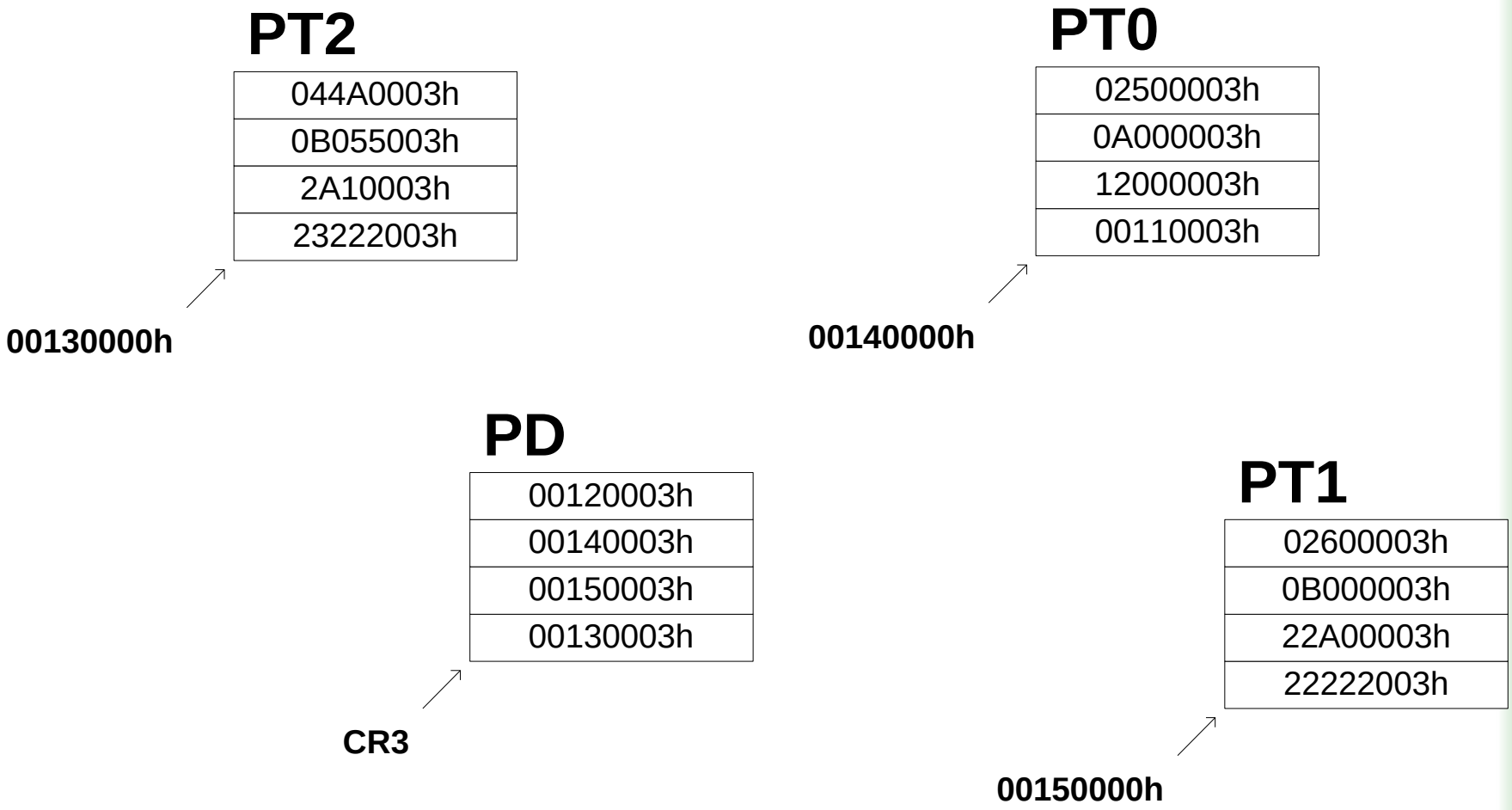




Paging Exercise

- Given is a system with one page directory PD and 3 page tables PT0-PT2.
  Control register CR3 holds the base address of the PD.
- Determine the effective physical addresses for the following linear addresses:
  - 00000000h
  - 00000FFFh
  - 00001ABCh
  - 00402321h
  - 00803EEEh

Paging Exercise



Paging Exercise Solution

Linear address = 00803EEEh

0 0 0 0 0 0 0 0 1 0	0 0 0 0 0 0 0 0 1 1	1 1 1 0 1 1 1 0 1 1 1 0
PDE = 2h	**PTE = 3h**	**Offset = 0EEEh**

PD

00120003h
00140003h
00150003h
00130003h

PDE = 2h → **Entry =** 00140003h

0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 1 1
PT base address = 00140000h	

00140000h points to PT0

PT0

02500003h
0A000003h
12000003h
00110003h

PTE = 3h → **Entry =** 02500003h

0 0 0 0 0 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 1 1
Page base address = 02500000h	

Page base address	02500000h
+ Offset	00000EEEh
Physical address	02500EEEh