MIT-WPU

॥ विश्वशान्तिर्ध्रुवं ध्रुवा ॥

Microprocessors and Microcontrollers

CS232 Microprocessors and Microcontrollers

Examination scheme:

Continuous Assessment: 50 Marks, End Semester: 50 Marks

Course Objectives:

1. To learn the architecture and programming of Pentium Microprocessor.
2. To study the protected memory management Mechanism.
3. To provide insight to multitasking environment.
4. To learn the architecture and programming of 8051 Microcontroller

Course Outcomes:

After completion of the course the students will be able to :

1. List and elaborate the advanced architectural features of Pentium processors and basic assembly language programming.
2. Develop programs using 80x86 assembly language.
3. Describe the theory of protected memory management.
4. Comprehend the concept of multitasking environment.
5. Illustrate architectural features and basic building blocks of 8051 microcontroller
6. Develop basic programs using 8051 assembly language.

Syllabus

Module I: Pentium Architecture :

- Pentium features, Operating Modes, Pentium super-scalar architecture - Pipelining, Branch prediction, Instruction and Data caches. Pentium programmer's model, Register set,System registers, Addressing modes and Instruction set. The Floating point Unit: features, pipeline stages & data types

Module II: Management in Protected Mode:

- Segmentation unit : Introduction, support registers, related instructions, segment Memory descriptors, logical to linear address translations, protection by segmentation, privilege-levels, rules of inter-privilege level transfer for data and code segments .

Syllabus

Module III: Paging and Task Management

- Paging Unit: support registers, related data structures,linear to physical address translation,TLB ,page level protection.
- Task Management -support registers, related data structures, Task switching.

Module IV: 8051 Microcontroller:

- Features, Micro-controller 8051 architecture. Programmers model-register set, register bank, SFRs, addressing modes, instruction set, Memory organization:on-chip and external memory components. Interrupt structure, Timers and their programming, Serial port and programming.

Syllabus

Text Books:

1. James Antonakos , “The Pentium Microprocessor” , 2004, Pearson Education ISBN – 81-7808-545-3.

2. K.J.Ayala " The 8051 Microcontroller " ISBN 9788131511053.

Reference Books:

1. Intel architecture software developer's manual volume 3.

2. Intel 8051 datasheet.

3. Pentium® Processor Family Developer’s Manual

Laboratory work

List of Assignments

1. Write an ALP to sort 8 bit numbers in ascending/descending order.
2. Write an ALP to calculate Mean of 5 numbers using NDP.
3. Write an ALP to simulate 'cp' command in Linux.
4. Write an ALP to display the contents of GDTR, IDTR, LDTR, TR and MSW .
5. Write an ALP to multiply 16 bit number by 8 bit number using 8051.
6. Write an 8051 ALP for rate generation using Timer by using
   a. Polling method
   b. ISR method

Module I: Pentium Architecture

- Pentium features, Operating Modes
- Pentium super-scalar architecture - Pipelining, Branch prediction, Instruction and Data caches.
- Pentium programmer's model, Register set, Addressing modes and Instruction set.
- The Floating point Unit: features, pipeline stages & data types.

Text Books & Reference books page numbers for Module I

Sr. No	Topic	Name of the Book	Page Number
1	Introduction and Evolution of Microprocessors, Pentium features, Operating Modes,	R4 R1	1-1 to 1-2 2-6 to 2-7
2	Pentium super-scalar architecture	T1	400
3	Pipelining, Branch prediction, Instruction and Data caches.	T1 R4	401 to 411 2-1 to 2-8
4	Pentium programmer's model, Register set, System registers, Addressing modes	R1 T1	2-1 to 2-6 and 2-8 to 2-18 70-81
5	Instruction set.	T1	44-50
6	The Floating point Unit: features, pipeline stages & data types.	T1 R4	411-414 2-9 to 2-11

T1: James Antonakos , “The Pentium Microprocessor” , 2004, Pearson Education ISBN – 81-7808-545-3.

T2: K.J.Ayala" The 8051 Microcontroller " ISBN 9788131511053.

R1: Intel architecture software developer's manual volume 3.

R4: Pentium® Processor Family Developer’s Manual

- T1: James Antonakos , “The Pentium Microprocessor” , 2004, Pearson Education ISBN – 81-7808-545-3.
- T2: K.J.Ayala" The 8051 Microcontroller " ISBN 9788131511053.
- R1: Intel architecture software developer's manual volume 3.
- R2: Intel architecture software developer's manual volume 1.
- R3: Intel 8051 datasheet.
- R4: Pentium® Processor FamilyDeveloper’s Manual
-

MIT-WPU
॥ विश्वशान्तिर्ध्रुवं ध्रुवा ॥

Historical Evolution

		Clock	
1971	4004	400 – 800 kHz	4-bit word size
1972	8008	500 – 800 kHz	8-bit word size
1974	8080	2 MHz	8 bit registers, 8 bit data bus, 16 bit address bus.
1978	8086 and related 8088	5MHz	16 bit registers, 16 bit data bus, 20 bit address bus. 6 bytes prefetch Q
1980	8087 floating–point coprocessor	Used with	8088, 8086, 80286, 80386, and 80486
1982	80286	6MHz	24 bit, 6 bytes prefetch Q
1983	80386	16-33MHz	32 bit, 16 bytes prefetch Q Three memory protection modes: **protected**, **real**, and **virtual**.

MIT-WPU
॥ विश्वशान्तिर्ध्रुवं ध्रुवा ॥

Historical Evolution

1989	80486	25-100MHz	32 bit,Pipelining,8 KB cache, floating point unit in the CPU core, 32 bytes prefetch Q
1993	Pentium	60-300MHz	Superscalar,128 bit registers performing 4 floating point computations in parallel, two 64 bytes prefetch Q
1995	Pentium Pro	166-200MHz	32 bit
1997	Pentium II	233-450MHz	32 bit
1999	Pentium III	450-1000MHz	32 bit
2001	Pentium 4	1400-3200MHz	144 instructions
Multicore Processors	I3,i5,i7		

Pentium features

- 66-100MHz Clock
- 32bit registers to hold data
- 32bit address bus
- 64bit data bus (burstable)
- Superscalar – 2 execution pipelines U and V
- Separate Code and Data Caches1.
  - 8KB 2-way set associative code cache + TLB
  - 8KB 2-way, dual access data cache + TLB
- 2 Prefetch buffers
- A Branch target Buffer to support Branch prediction logic
- Dynamic Branch Prediction
- Pipelined Floating-Point Unit
- Improved Instruction Execution Time
- Write back MESI Protocol in the Data Cache

Operating Modes

- **Real-address mode:**

  This operating mode provides the programming environment of the Intel 8086 processor, with a few extensions (such as the ability to switch to protected or system management mode).
  - Uses 16-bit addresses
  - Runs 8086 programs
  - Pentium acts as a faster 8086

- **Virtual-8086 mode:**

  In protected mode, the processor supports a quasi-operating mode known as virtual-8086 mode. This mode allows the processor to execute 8086 software in a protected, multitasking environment.

Continued…

- **Protected mode:**

  This is the native operating mode of the processor. In this mode all instructions and architectural features are available, providing the highest performance and capability. This is the recommended mode for all new applications and operating systems.
  - 32-bit mode
  - Native mode of Pentium
  - Supports segmentation and paging

- **System management mode (SMM):**

  This mode provides an operating system or executive with a transparent mechanism for implementing power management and OEM differentiation features.

  SMM is entered through activation of an external system interrupt pin (SMI#), which generates a system management interrupt (SMI).

  In SMM, the processor switches to a separate address space while saving the context of the currently running program or task.

  SMM-specific code may then be executed transparently. Upon returning from SMM, the processor is placed back into its state prior to the SMI.

Continued…

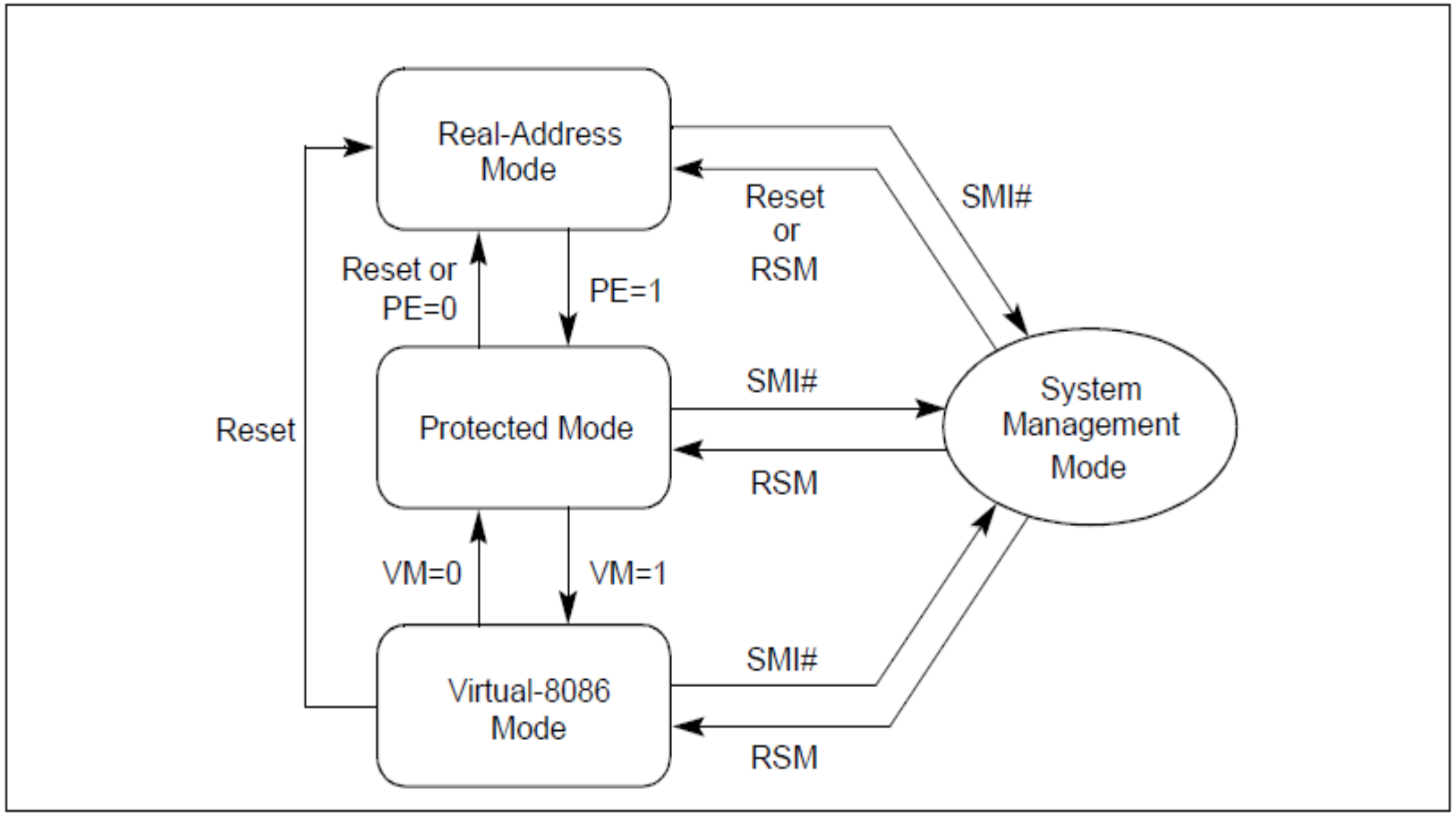


Figure 2-2. Transitions Among the Processor's Operating Modes

- SMM is a special-purpose operating mode provided for handling system-wide functions like **power management, system hardware control**, or proprietary OEM designed code.
- It is intended for use only by system firmware, not by applications software or general-purpose systems software.
- The main benefit of SMM is that it offers a distinct and easily isolated processor environment that operates transparently to the operating system or executive and software applications.

Continued…

- The processor is placed in real-address mode following power-up or a reset.
- Thereafter, the PE flag in control register CR0 controls whether the processor is operating in real-address or protected mode.
- The VM flag in the EFLAGS register determines whether the processor is operating in protected mode or virtual-8086 mode.
- Transitions between protected mode and virtual-8086 mode are generally carried out as part of a task switch or a return from an interrupt or exception handler.
- The processor switches to SMM whenever it receives an SMI while the processor is in real address,protected, or virtual-8086 modes.
- Upon execution of the RSM instruction, the processor always returns to the mode it was in when the SMI occurred.

Pentium Architecture

- It has data bus of 64 bit and address bus of 32- bit
- There are two separate 8kB caches – one for code and one for data.
- Each cache has a separate address translation TLB which translates linear addresses to physical.
- Code Cache: – 2 way set associative cache – 256 lines b/w code cache and prefetch buffer, permitting prefetching of 32 bytes (256/8) of instructions

Architecture

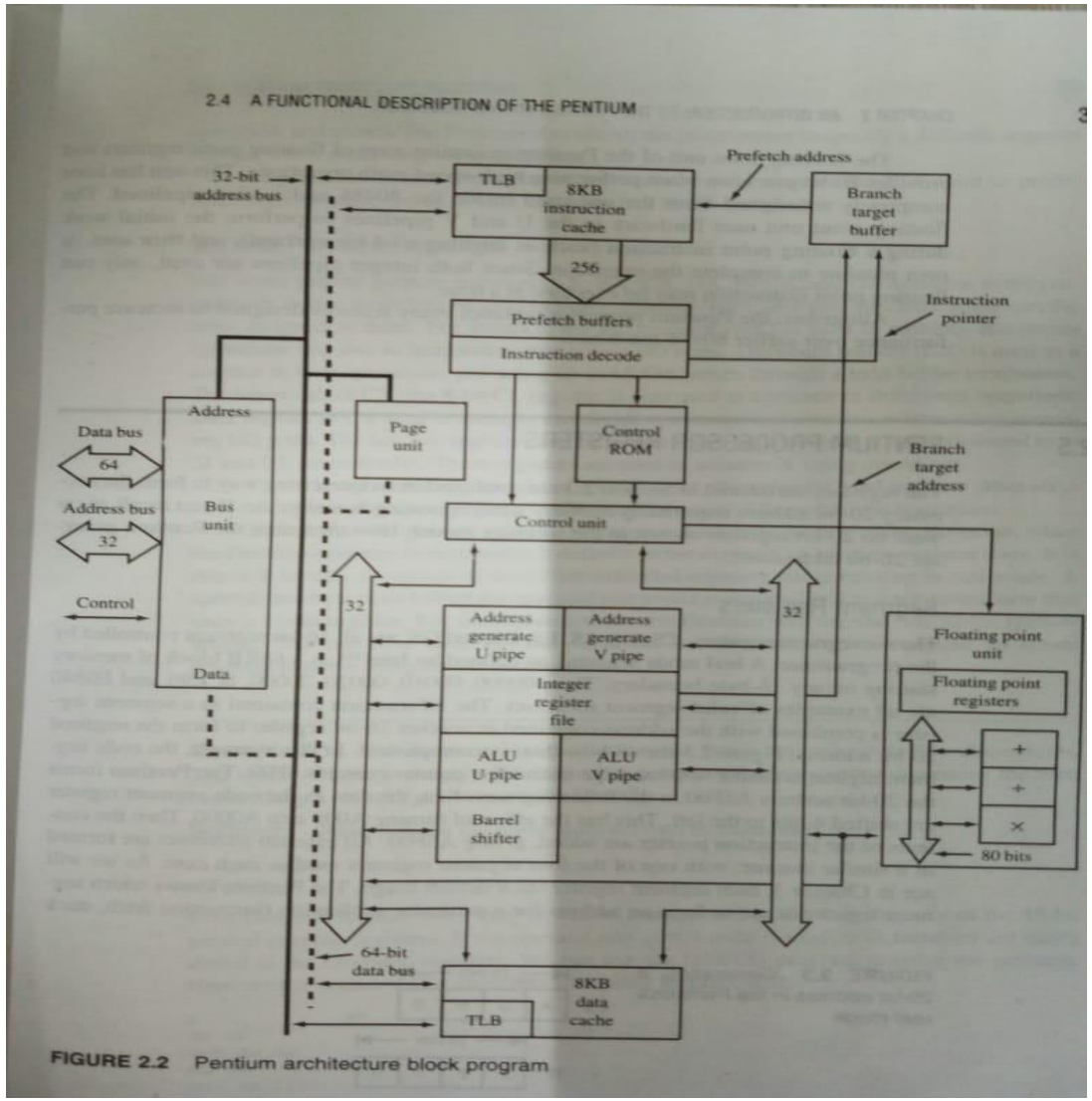
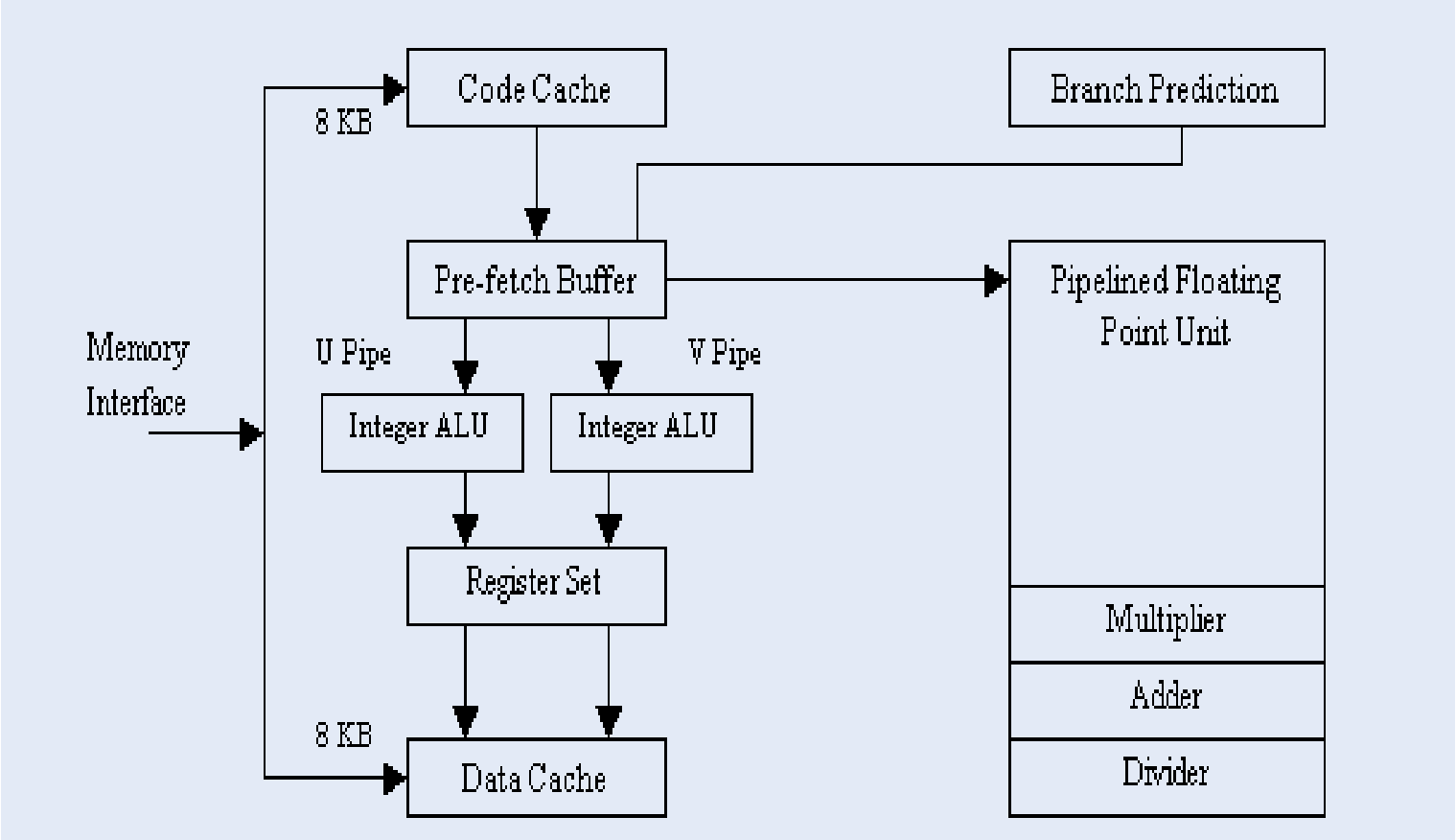


FIGURE 2.2 Pentium architecture block program

Superscalar Architecture



- Prefetch Buffers: 2 prefetch buffers
  - When instructions are prefetched from cache, they are placed into one set of prefetch buffers.
  - The other set is used as when a branch operation is predicted.

- Instruction Decode Unit:

  It occurs in two stages – Decode1 (D1) and Decode2(D2)
  - D1 checks whether instructions can be paired
  - D2 calculates the address of memory resident operands

Control Unit :

- It interprets the instruction word and microcode entry point fed to it by Instruction Decode Unit
- It handles exceptions, breakpoints and interrupts.
- It controls the integer pipelines and floating point sequences
- Microcode ROM : Stores microcode sequences

Arithmetic/Logic Units (ALUs) :

There are two parallel integer instruction pipelines:

u- pipeline and v-pipeline

- The u-pipe can execute all integer and floating-point instructions.
- The v-pipe can execute simple integer instructions and the FXCH floating-point instruction.

RISC & CISC

Pentium uses both RISC and CISC Characteristics.

RISC has the advantages of

- small instructions,
- fewer instructions ,
- simple addressing modes,
- Make good use of registers and pipeline everything.

The CISC attributes are

- Instructions with variable length
- Instructions with variable execution time
- Instructions using different addressing modes
- Instruction set includes complex instructions such as MUL, DIV, XCHG
- Instructions executed using microcode

To keep compatibility with earlier processors, CISC is used.

MIT-WPU
।। विश्वशान्तिर्ध्रुवं ध्रुवा ।।

Pentium **super-scalar** architecture

- Pentium is capable of executing 2 integer or 2 floating point instructions simultaneously.
- This parallel execution is done through two instruction pipelines, the "u" pipe and the "v" pipe.
- The u-pipe can execute all integer and floating-point instructions.
- The v-pipe can execute simple integer instructions and the FXCH floating-point instruction.
- Processors capable of parallel instruction execution of multiple instructions are known as Superscalar machines.

Instruction Pairing Restrictions (for parallel execution)

1. Both must be simple instructions.

2. No data dependencies may exist between them.

   E.g. Add AX,BX
   Add AX,CX
   For floating point instructions,
   The first instruction of the pair can be – FLD, FLD st(i), FADD, FMUL, FDIV, FCOM, FUCOM, FTST, FABS, FCHS.
   The second instruction must be FXCH .

3. Neither instruction may contain both immediate data and a displacement value (MOV TABLE[SI],7).

4. Prefixed instructions may execute only in U pipeline (MOV ES:[DI],AL).

Cycle1⇒ FADD QWORD PTR [EAX] FXCH ST(2)
Cycle 2⇒ FNUL QWORD PTR [EBX] FXCH ST(3)

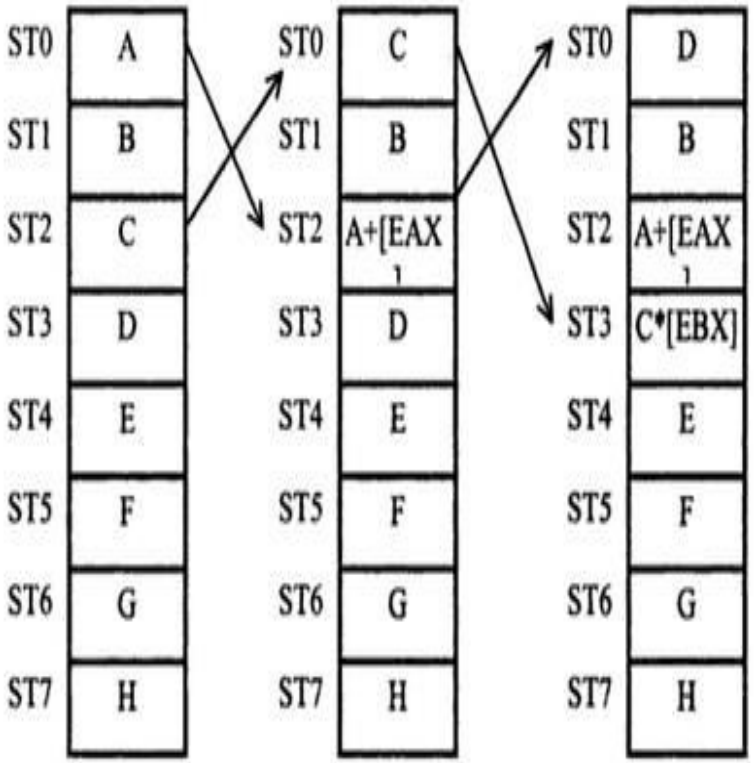


Fig.4.8 *(a)* FXCH code example

The code in the example generates the results of 2-independent floating-point calculations. The floating-point register file contains initial value prior to code execution

- register STO (TOS) contains the value A,
- register ST1 contains the value B,
- register ST2 contains value C, and so on.

The 2-operations are

- floating-point addition of value A with the 64-bit floating- point operand addressed by the general register EAX, and
- floating-point multiplication of value C by the 64-bit floating-point operand addressed by the general register EBX.

When the floating-point pipeline is fully loaded and these 2-operations are part of the code sequence, the parallel FXCH allows the calculation to maintain maximum throughput of one cycle per operation. Within one cycle the Pentium μP writes the result of the addition to ST2, while the operand for the next operation moves to the top of the stack. in the next cycle, the processor writes the result of the multiplication to ST3, while the top of the stack contains value D, which may be used for a subsequent operation.

Continued…

Pentium has U and V pipeline for parallel execution

- U pipeline : capable of handling full instruction set
- V pipeline : only simple instructions

Simple Instructions

- are hardwired and do not need microcode, execute in one clock cycle

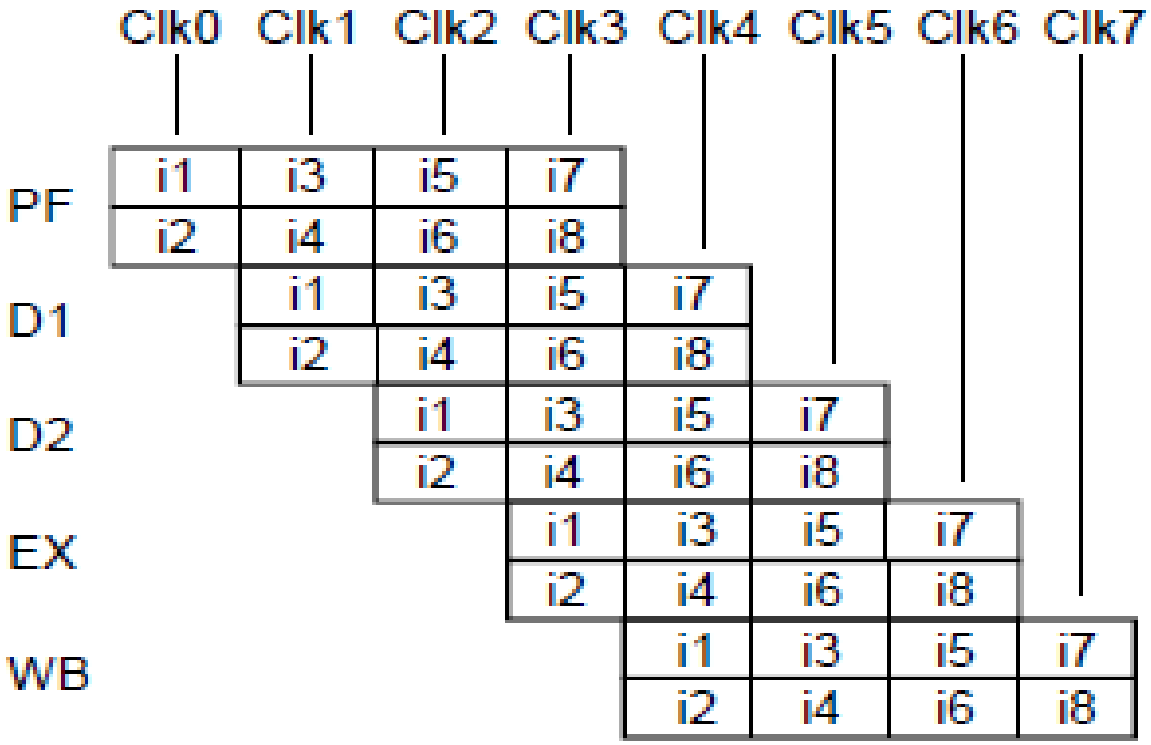
1. mov reg, reg/mem/imm
2. mov mem, reg/imm
3. alu reg, reg/mem/imm
4. alu mem, reg/imm
5 inc reg/mem
6. dec reg/mem
7. push reg/mem
8. pop reg
9. lea reg,mem
10. jmp/call/jcc near
11. nop
12. test reg, reg/mem
13. test acc, imm

Pipelining

The pipeline stages in the Pentium processor are as follows:

1. Pre fetch: instructions are pre fetched from the on-chip instruction cache or memory.
2. Decode1:The decoders decodes instructions and decides whether they can be paired
3. Decode2: addresses of memory resident operands are calculated.
4. Execute: Execute the instructions with Cache and ALU access
1. Write Back: modify processor state and complete execution.

Pipelining


Clk0
Clk1
Clk2
Clk3
Clk4
Clk5
Clk6
Clk7
PF
D1
D2
EX
WB
i1
i3
i5
i7
i2
i4
i6
i8

Pipeline performance

- The maximum throughput is observed with-
  - Every instruction takes exactly one clock cycle to execute
  - No or minimum Stalls in pipeline

The number of clock cycles = m+(n-1) where

m = number of stages in a pipeline

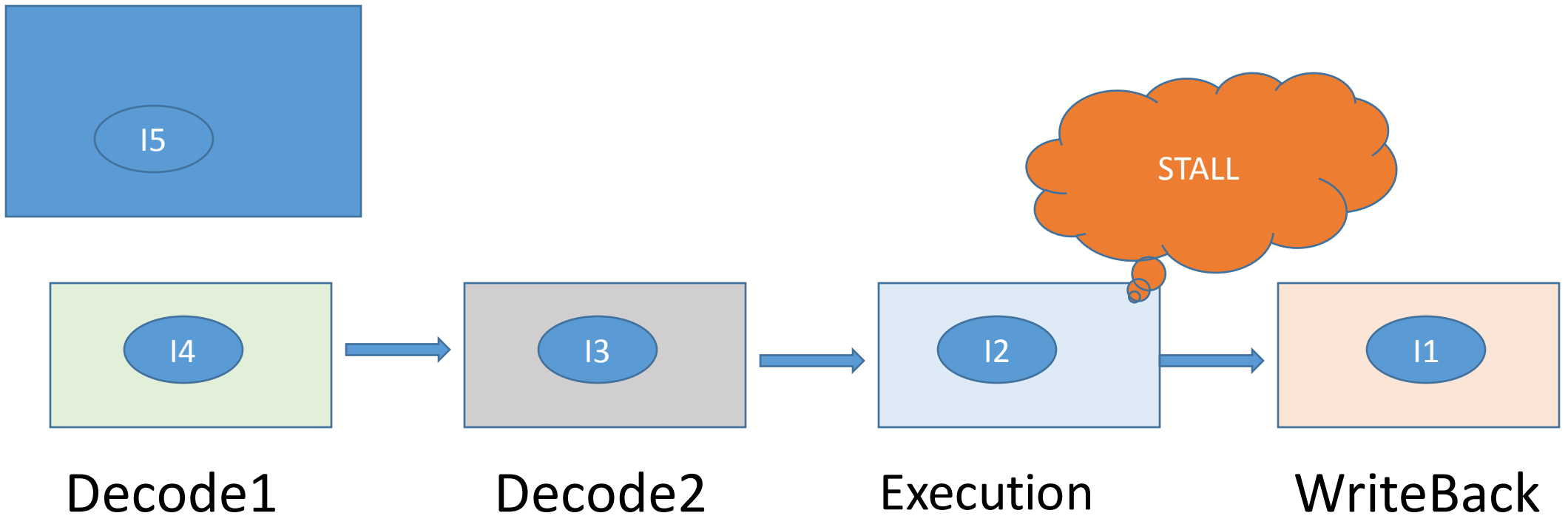
n = Number of instructions in a program

E.g. a sequence of 1000 instructions will require 3000 clk cycles on a non pipelined machine and only 1002 cycles when pipelined!

Execution Unit - creating Stall / bubble

I1, I3, I4 : Simple instructions : 1 CLK Cycle

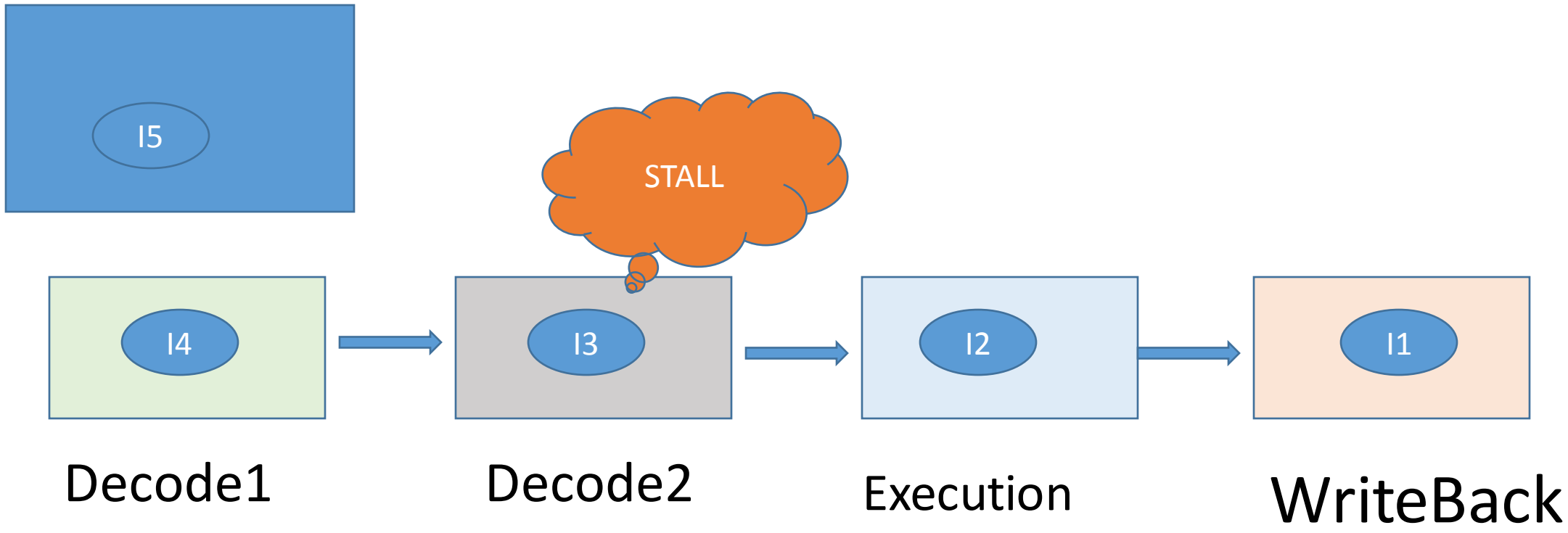
I2 : Multiplication Instruction : 4CLK Cycles



Decode2 : Operand fetch, Address generation - creating stall

I1, I2, I4 : Simple instructions : 1 CLK Cycle

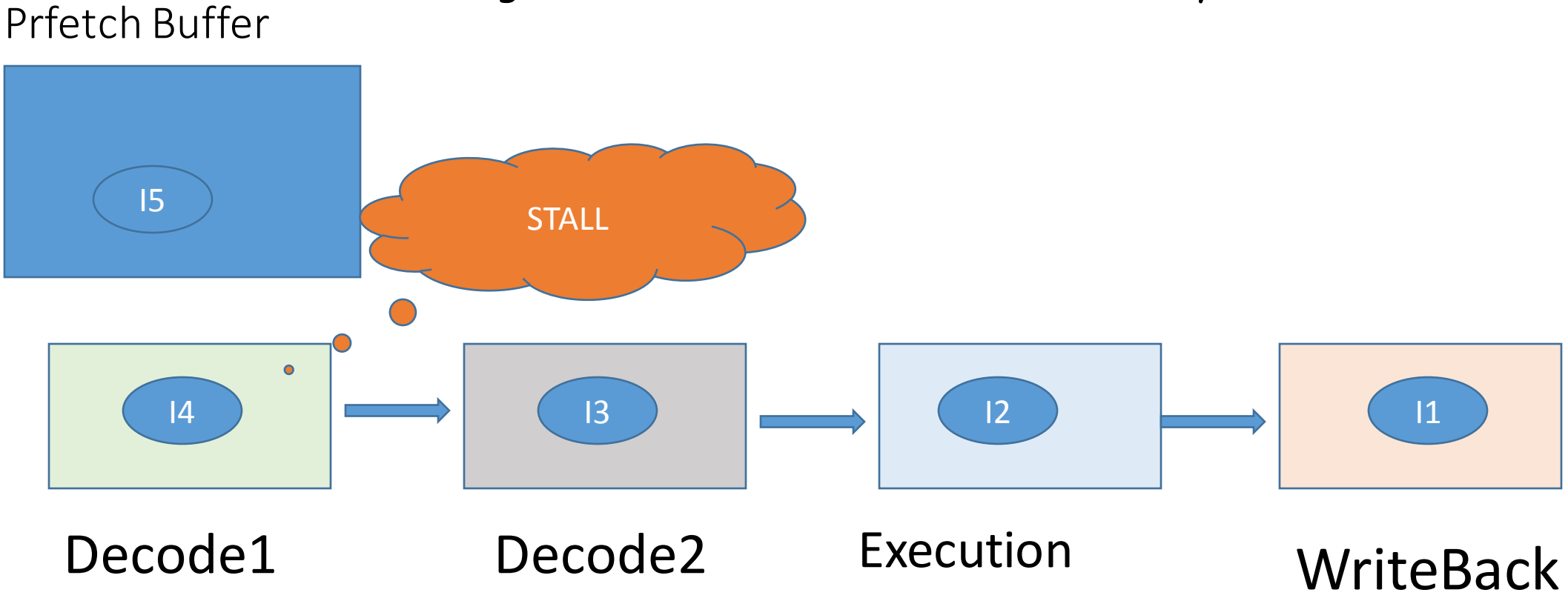
I3 : mov ax, buffer[bx+si] : 4CLK Cycles



Decode1 : Instr. Decoding, instruction pairing – creating stall

I1, I2, I3 : Simple instructions : 1 CLK Cycle

I4 : Xchg : 4CLK Cycles



Pipelining stages of Pentium

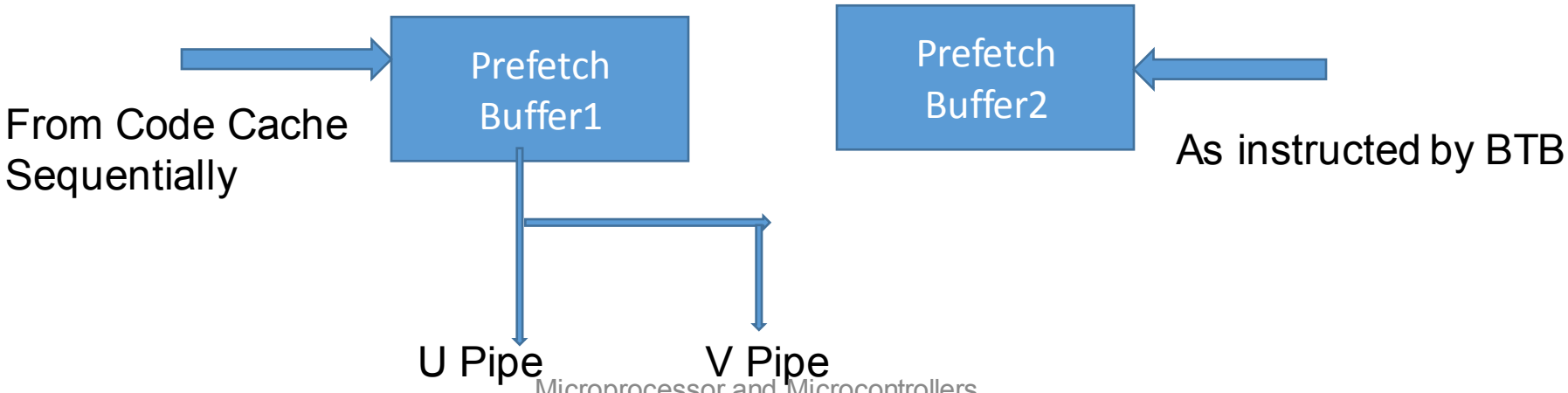
1. The first stage of the pipeline is the Prefetch (PF) stage in which instructions are prefetched from the on-chip instruction cache or memory.

- Because the Pentium processor has separate caches for instructions and data, prefetches do not conflict with data references for access to the cache.
- In the PF stage ,two independent pairs of 32-byte prefetch buffers operate in conjunction with the branch target buffer.
- This allows one prefetch buffer to prefetch instructions sequentially, while the other prefetches according to the branch target buffer predictions.

Prefetch stage

- Two pre fetch buffers, 32bytes each.
- At a time only one pre fetch buffer is connected to u and v pipes.

- **The buffer connected to the pipes-**
  - fetches instructions from code cache sequentially.
- **The other prefetch buffer**
  - fetches the instructions as directed by BTB.



Continued…

2. The second stage is Decode1 (D1) in which two parallel decoders attempt to decode and issue the next two sequential instructions.

The decoders determine whether one or two instructions can be issued(the instruction pairing rules).

3. The D1 stage is followed by Decode2 (D2) in which addresses of memory resident operands are calculated.

Continued…

4. The Pentium processor uses the Execute (EX) stage of the pipeline for both ALU operations and for data cache access.

- In EX all u-pipe instructions and all v-pipe instructions except conditional branches are verified for correct branch prediction.
- Microcode is designed to utilize both pipelines and thus those instructions requiring microcode execute faster.

5. The final stage is Writeback (WB) where instructions are enabled to modify processor state and complete execution.

- In this stage, v-pipe conditional branches are verified for correct branch prediction.

MIT-WPU
।। विश्वशान्तिर्ध्रुवं ध्रुवा ।।

Continued…

- Both the u-pipe and v-pipe instructions enter and leave the D1 and D2 stages simultaneously.
- When an instruction in one pipe is stalled, then the instruction in the other pipe is also stalled at the same pipeline stage.
- Thus both the u-pipe and the v-pipe instructions enter the EX stage simultaneously.
- Once in EX if the u-pipe instruction is stalled, then the v-pipe instruction (if any) is also stalled.
- If the v-pipe instruction is stalled then the instruction paired with it in the u-pipe is not allowed to advance.
- No successive instructions are allowed to enter the EX stage of either pipeline until the instructions in both pipelines have advanced to WB.

Branch Prediction

- Performance gain through pipelining can be reduced by the presence of program transfer instructions (such as JMP, CALL, RET and conditional jumps).
- They change the sequence causing all the instructions that entered the pipeline after program transfer instruction invalid.
- Suppose instruction I3 is a conditional jump to I50 at some other address (target address), then the instructions that entered after I3 is invalid and new sequence beginning with I50 need to be loaded in.
- This causes bubbles in pipeline, where no work is done as the pipeline stages are reloaded.

Continued…

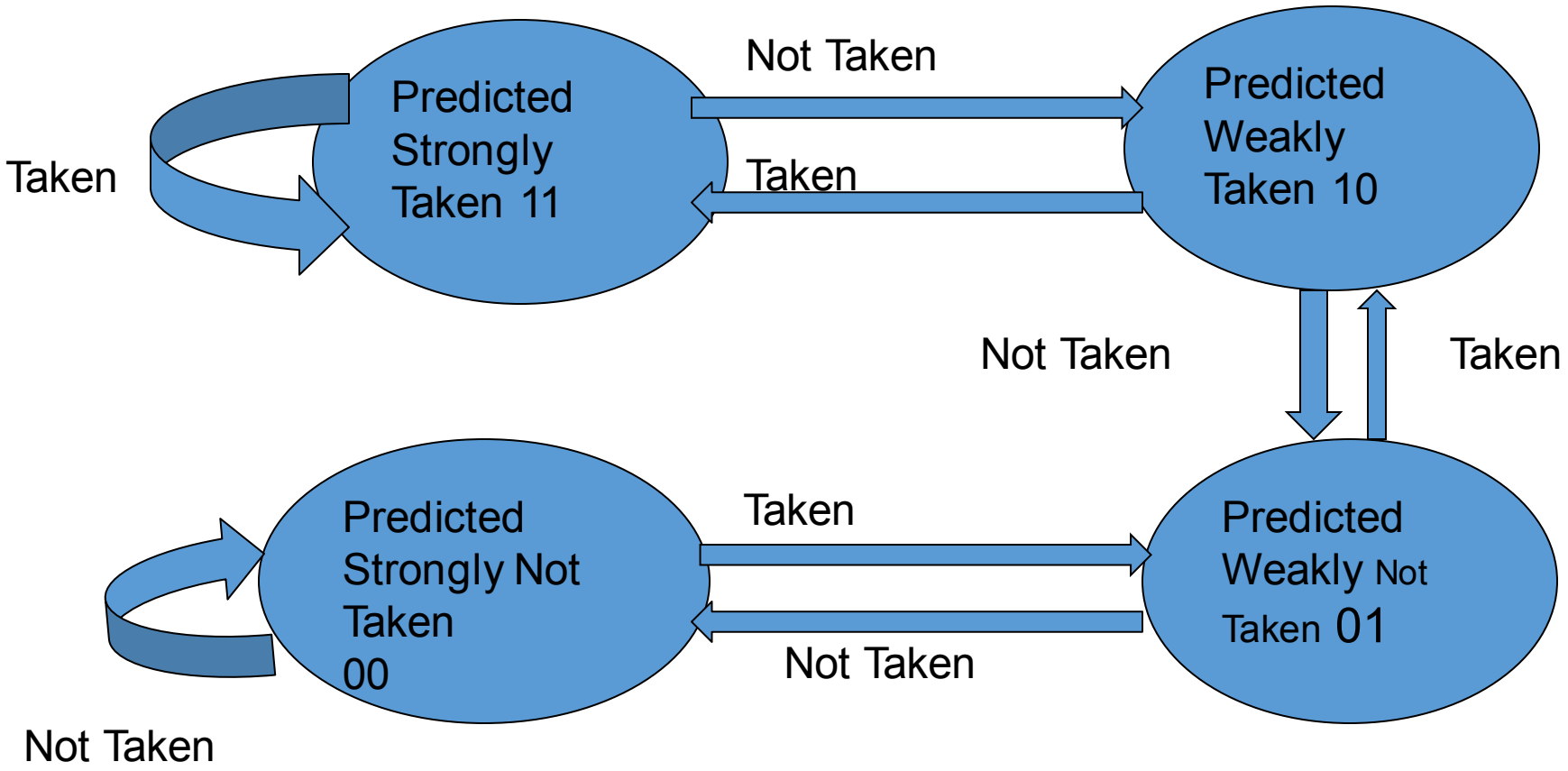
- In this scheme, a prediction is made concerning the branch instruction currently in pipeline.
- Prediction will be either taken or not taken.
- If the prediction turns out to be true, the pipeline will not be flushed and no clock cycles will be lost.
- If the prediction turns out to be false, the pipeline is flushed and started over with the correct instruction.
- To avoid this problem, the Pentium uses a scheme called Dynamic Branch Prediction.

Branch Prediction

- The Pentium processor uses a Branch Target Buffer (BTB) to predict the outcome of branch instructions which minimizes pipeline stalls due to prefetch delays.

- **Algorithm implemented in Pentium for reducing pipeline stalls during execution of jmp instructions**

Branch Prediction Algorithm



Branch Prediction Algorithm

- Two 32 bytes prefetch buffers work with BTB.
- One fetches instruction from the current program address and the other buffer is activated when the branch BTB predicts “Taken”.
- Initially History bits are 11 when a new target address is placed in BTB.
- Whenever branch instruction is encountered, these bits are updated.
- If the branch is not taken repetitively then History bits become 00 and the prediction is “Not taken”
- As long as they are not both zero, the prediction is “ taken”.
- BTB is accessed with the address instruction during D1 stage.
- If found and the BTB’s prediction is “Taken”, second prefetch buffer becomes active.
- If a new branch instruction is encountered(no target address in BTB),the prediction is “Not taken”.

MIT-WPU
॥ विश्वशान्तिर्ध्रुवं ध्रुवा ॥

Example -

```
      I0    MOV SI, addr
Bk :  I1    INC SI
      I2    DEC CX
      I3    JNZ  BK
      I4    ADD  AX,BX
      I5    INC  AX
      I6    MOV  [SI] , AX
```

- During D1 stage ,

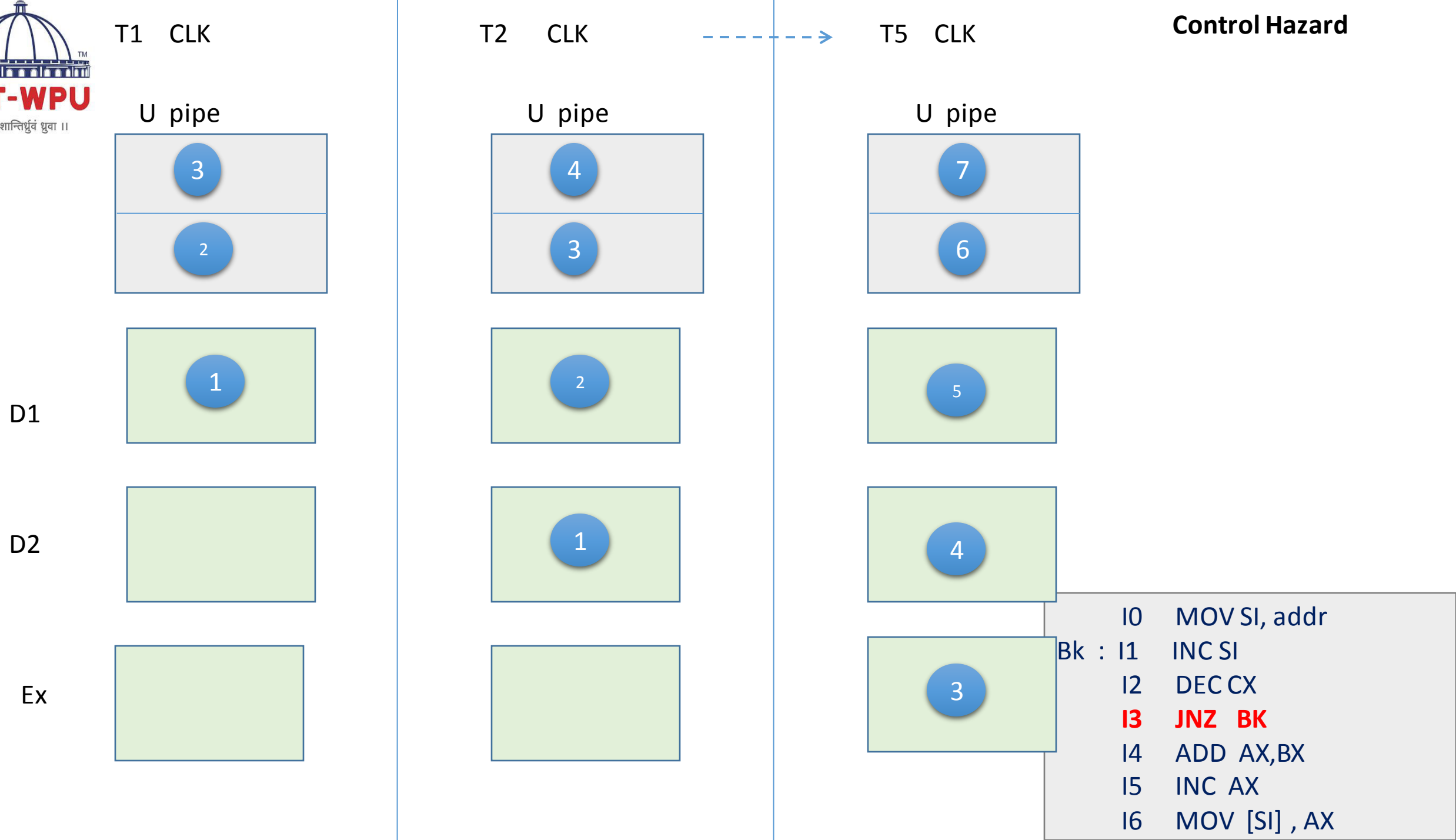
  the JNZ instruction is not known to BTB so it's a miss

  thus predicts

  NO JUMP WOULD BE TAKEN

MIT-WPU
॥ विश्वशान्तिर्ध्रुवं ध्रुवा ॥

Control Hazard

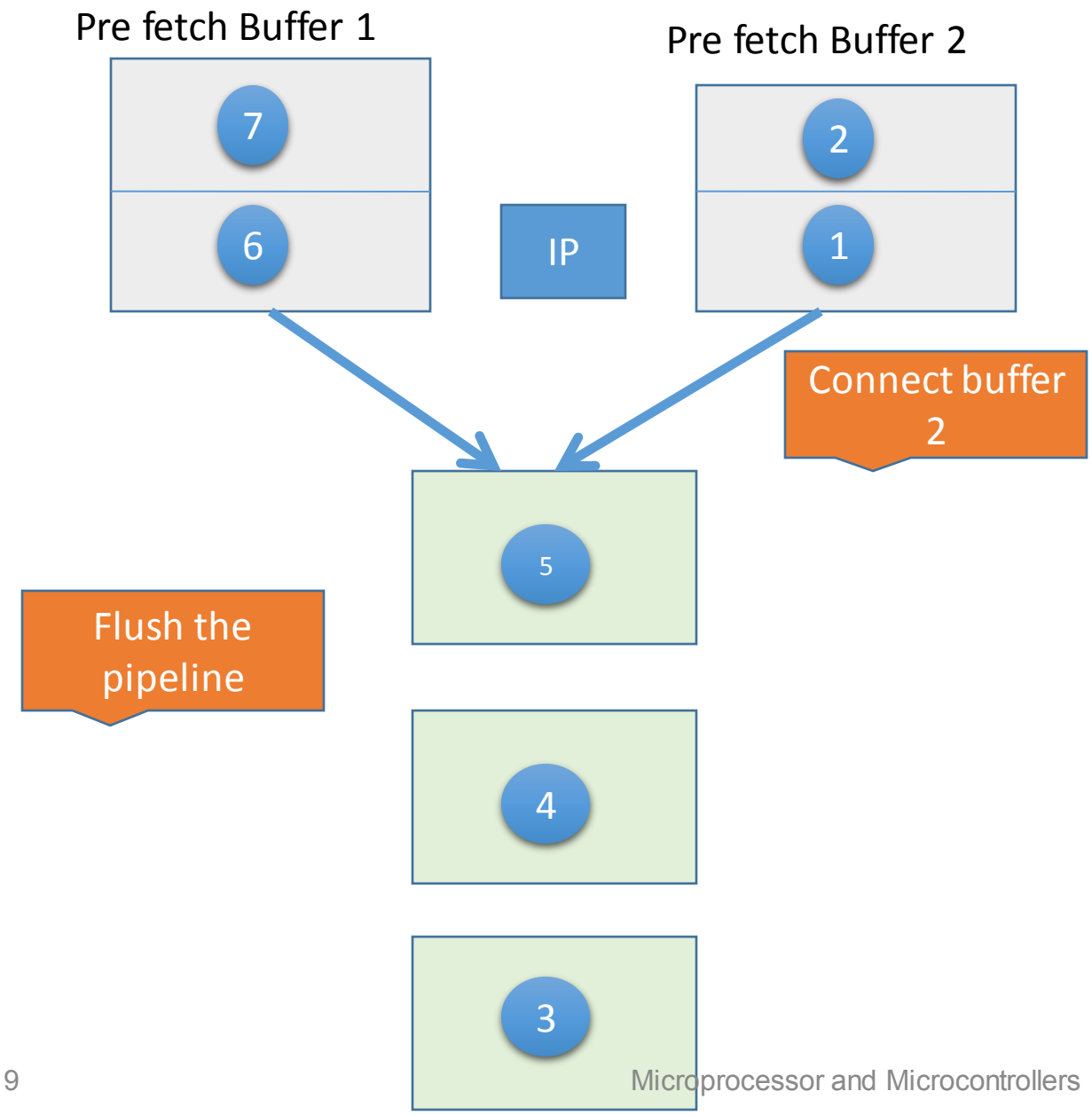


- The Execute stage gives feedback to BTB – Branch actually to be taken , and the BTB makes the entry as –

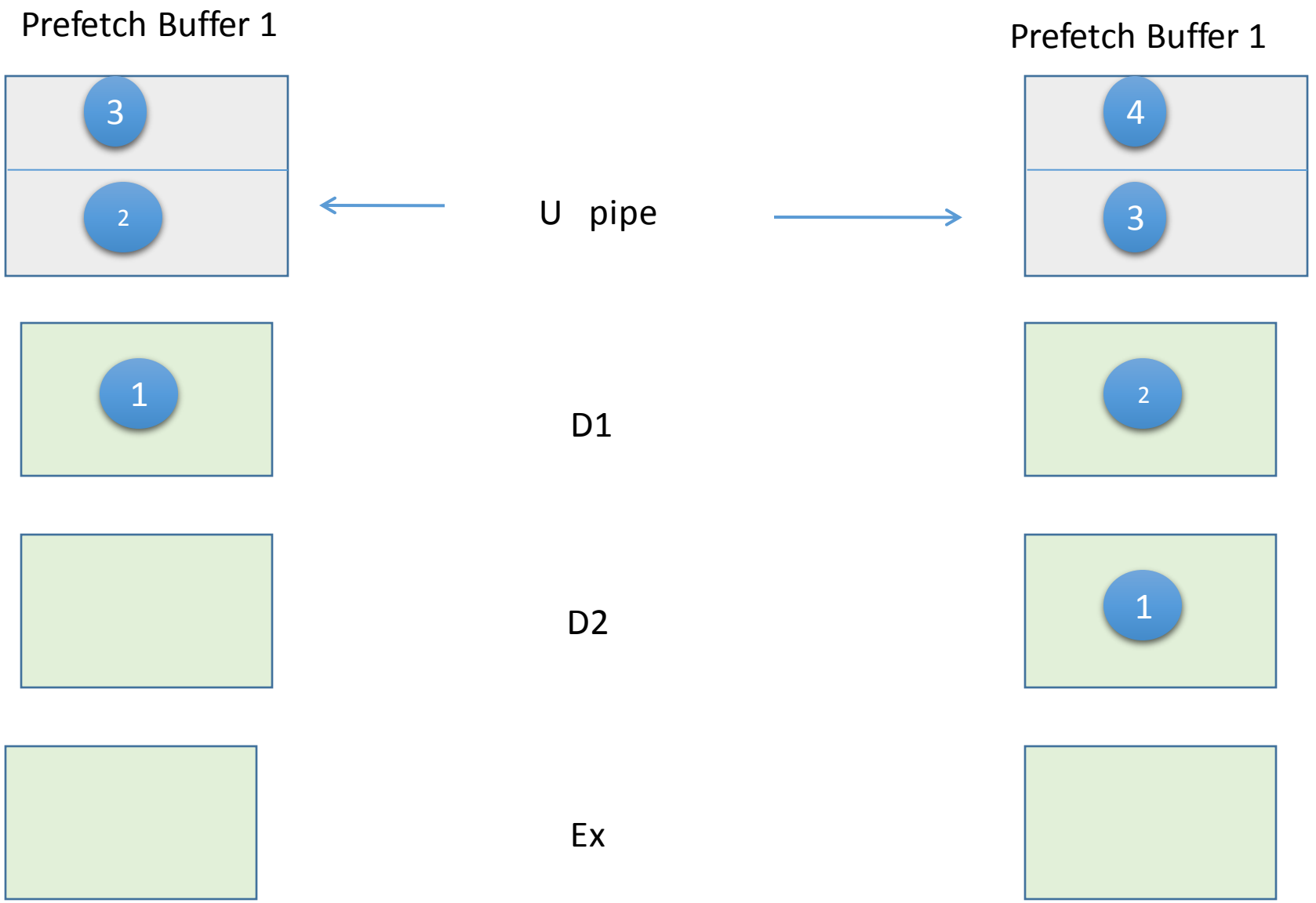
VALID BIT	HISTORY BITS	SOURCE ADDRESS	TARGET/BRANCH ADDRESS
1	11	Addr I3	Addr BK

- Pipeline flushed
- Instructions fetched from new address “BK” in the pre fetch buffer2
- Pre fetch buffer 2 connected to the pipeline

First occurrence of JNZ



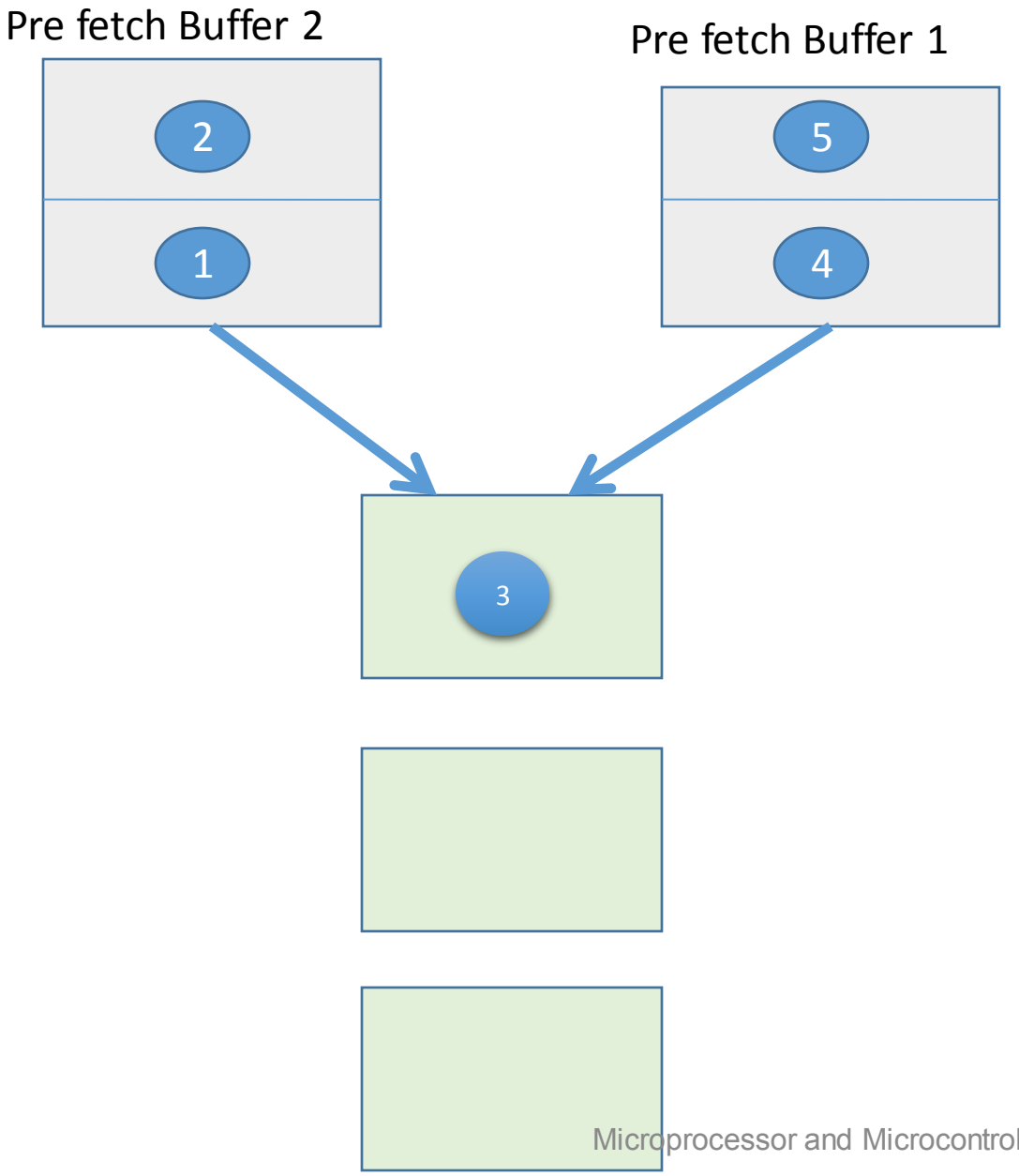

```
      I0    MOV SI, addr
Bk :  I1    INC SI
      I2    DEC CX
      I3    JNZ  BK
      I4    ADD  AX,BX
      I5    INC  AX
      I6    MOV  [SI] , AX
```


Prefetch Buffer 1
3
2
U pipe
Prefetch Buffer 1
4
3
1
D1
2
D2
1
Ex

Second occurrence of “JNZ” instruction

- BTB Hit, Prediction – Branch Strongly taken when reaches Execute Stage
- BTB instructs prefetch stage to fetch instructions from addr. “BK”
- The new instructions are filled in prefetch buffer2
- PF buffer1 disconnected while Buffer 2 connected to U and V pipe.

Second occurrence of JNZ




```
      I0    MOV SI, addr
Bk :  I1    INC SI
      I2    DEC CX
      I3    JNZ  BK
      I4    ADD  AX,BX
      I5    INC  AX
      I6    MOV  [SI] , AX
```

BTB Example

Consider the following loop for computing prime numbers:

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

A popular compiler generates the following assembly code:

(prime is allocated to ecx, k is allocated to edx, and al contains the value FALSE)

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

Each iteration of this loop will execute in 6 clocks on the Intel486 CPU.

On the Pentiump rocessor, the mov is paired with the add; the cmp with the jle. With branch prediction,each loop iteration executes in 2 clocks.

Data and Instruction Cache

- Code / Data separate Cache
- Two Level Cache – L1, L2

Common Attributes –

- Size - - 8KB
- Mapping Technique - 2way set associative
- Line Size – - 32bytes
- Type - Wr Back / Wr Thru
- Coherence Protocol - MESI

More about cache ….

- Pentium has 2 separate Data & Code(Instruction)cache .
- They are two way set associative caches with 128 sets. Total 256 entries.
- There are 32 bytes in a line(64 bytes per set):

  8 KB (128 x 64)storage.
- Both caches can be accessed simultaneously
- LRU algorithm is used to select entries in cache.
- The respective tags are triple ported(they can be accessed from 3 different places at the same time)
  - One port used for bus snooping(used to help maintain consistent data in a multiprocessor system)
  - The other two used for U and V pipelines

Cache

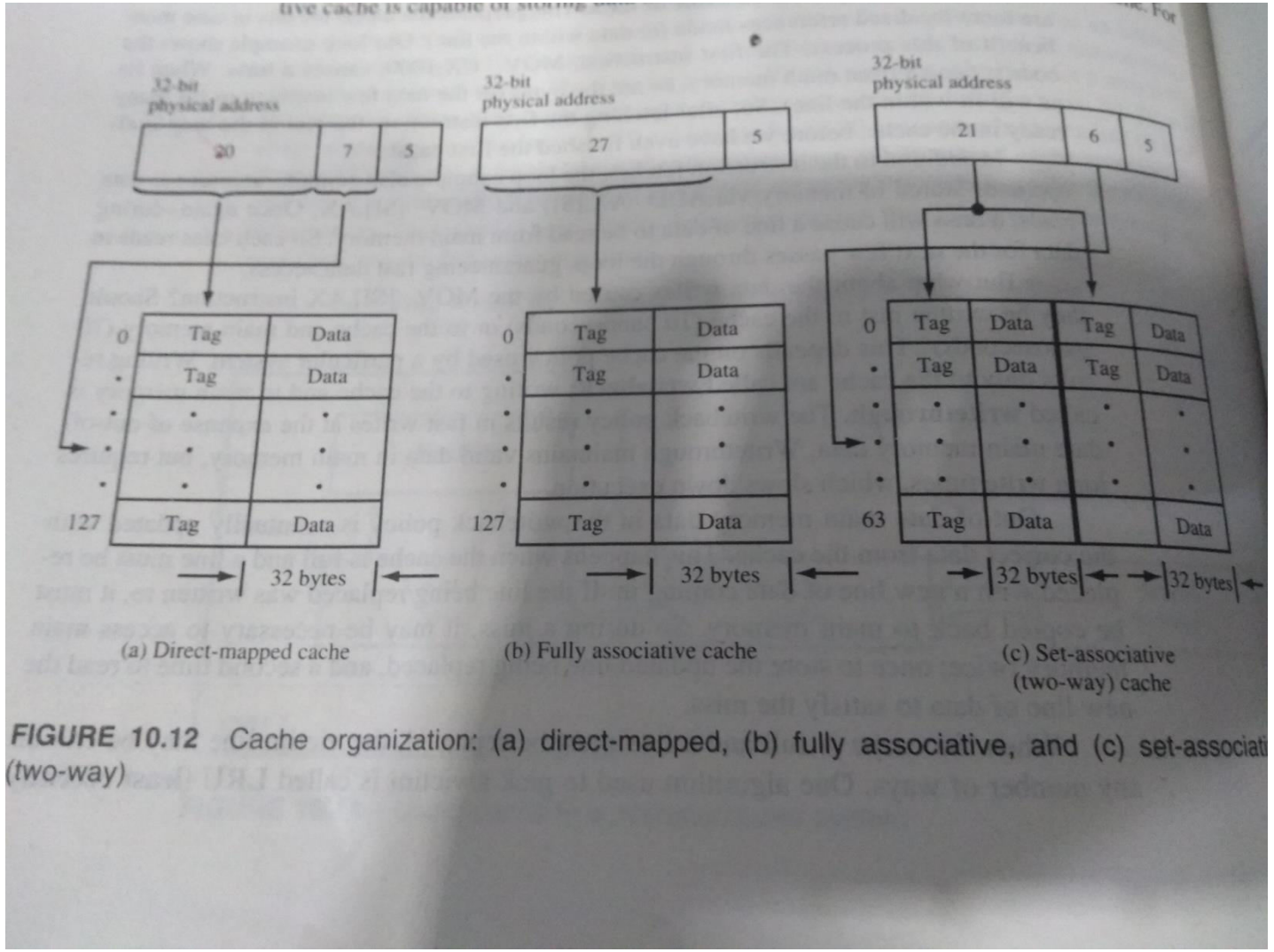


FIGURE 10.12 Cache organization: (a) direct-mapped, (b) fully associative, and (c) set-associati
(two-way)

More about cache ….

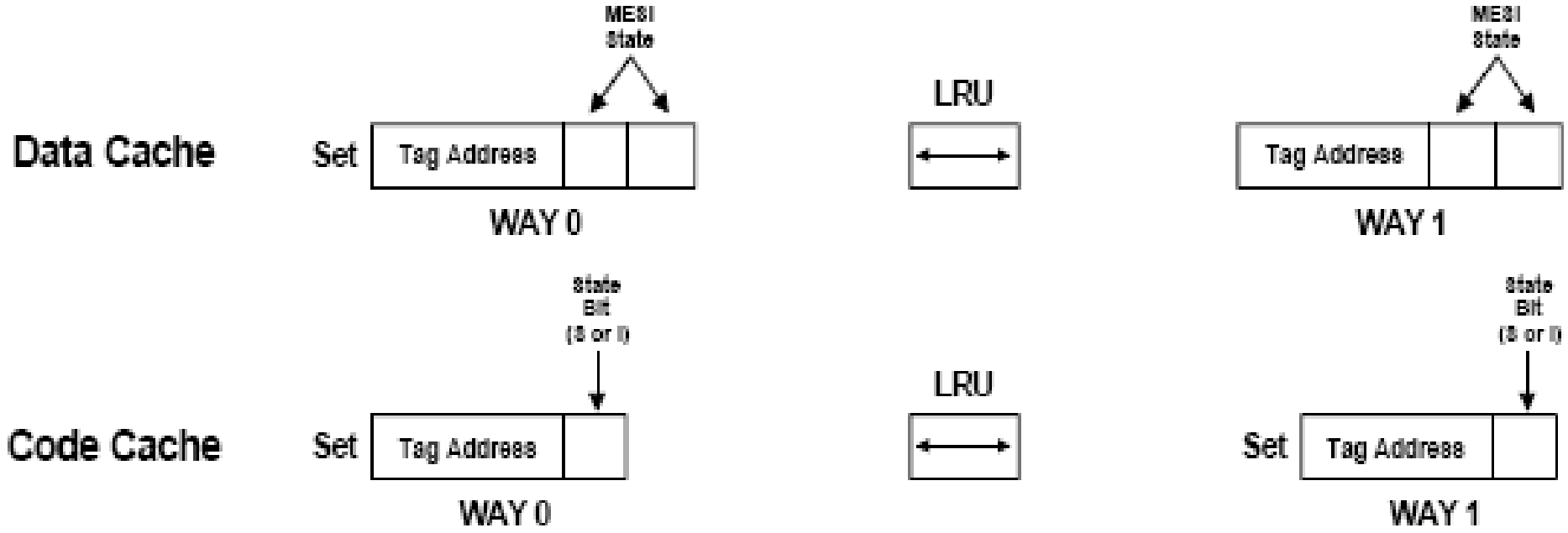
- Each entry in the data cache can be configured for writethrough (writing to cache and to the main memory) or writeback(writing only to cache)

- The data cache fully supports the MESI

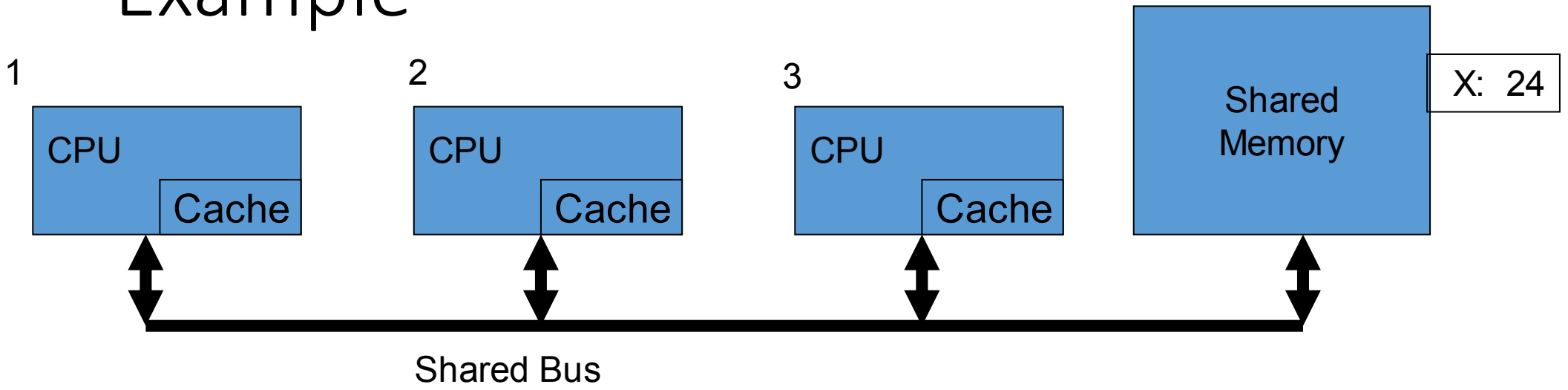
(modified/exclusive/shared/invalid) writeback cache consistency protocol.

- The code cache is inherently write protected to prevent code from being inadvertently corrupted, and as a consequence supports a subset of the MESI protocol, the S (shared) and I (invalid) states.

Code & Data cache organisation



Example



Processor 1 reads X: obtains 24 from memory and caches it
Processor 2 reads X: obtains 24 from memory and caches it
Processor 1 writes 32 to X: its locally cached copy is updated
Processor 3 reads X: what value should it get?
Memory and processor 2 think it is 24
Processor 1 thinks it is 32

MESI Protocol (2)

Any cache line can be in one of 4 states (2 bits)

- **Modified** - cache line has been modified, is different from main memory - is the only cached copy. (multiprocessor 'dirty')
- **Exclusive** - cache line is the same as main memory and is the only cached copy
- **Shared** - Same as main memory but copies may exist in other caches.
- **Invalid** - Line data is not valid (as in simple cache)

TLB - Translation Look Aside Buffer

- Each of the caches are accessed with physical addresses and each cache has its own TLB

- TLB translates linear address to physical address as required for paging technique

Floating Point Unit

- The floating-point unit (FPU) of the Pentium processor is integrated with the integer unit on the same chip.
- It is heavily pipelined.
- The Pentium processor FPU is implemented with 8 pipeline stages.
- Of these first 5 stages are shared with Integer Unit.
- The integer instruction uses the fifth stage as writeback.

MIT-WPU
॥ विश्वशान्तिर्ध्रुवं ध्रुवा ॥

FP instructions on the pipeline

- FP instruction does not get paired with integer instruction

- Two FP instructions can be paired only if
  a. FXCH instruction is the second instruction in the pair.
  b. The first instruction of the pair can be – FLD, FLD st(i), All forms of FADD, FMUL, FDIV, FCOM, FUCOM, FTST, FABS, FCHS
  c. The instructions which do not follow a and b always are issued to “u” pipe singly.

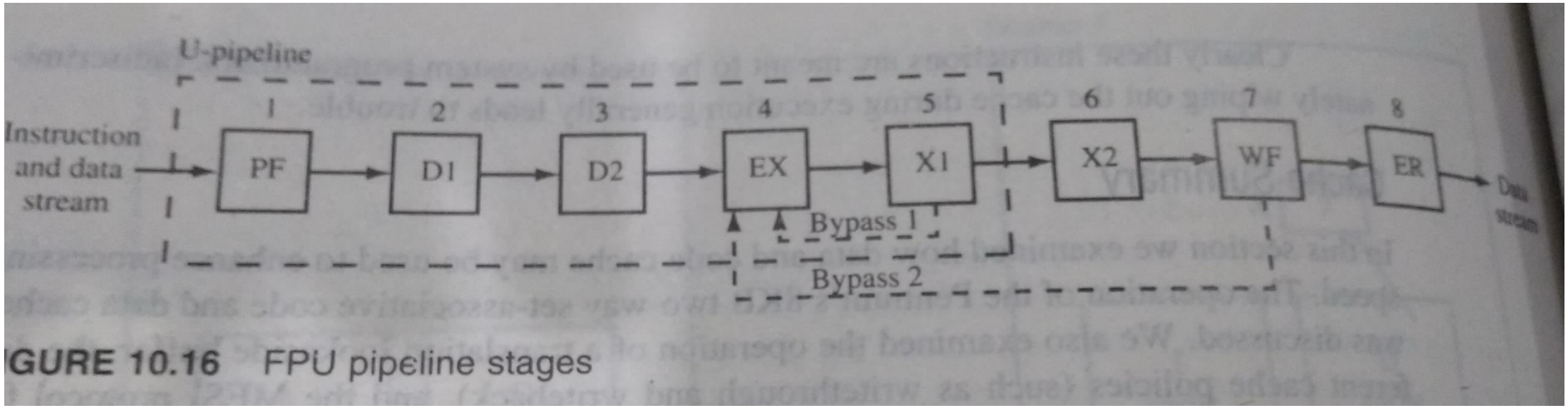
8 pipeline stages of FPU

1. PF Prefetch;
2. D1 Instruction Decode
3. D2 Address generation
4. EX Memory and register read : conversion of FP data to external memory format and memory write
5. X1 Floating-Point Execute stage one : conversion of external memory format to internal FP data format and write operand to FP register file; bypass 1(send data back to EX stage)
6. X2 Floating-Point Execute stage two;
7. WF Perform rounding and write floating-point result to register file; bypass 2 (send data back to EX stage)
8. ER Error Reporting/Update Status Word.

Bypass -

- With bypass technique,

  when the result from the previous instruction is about to be written to the register file, is made directly available as an operand to any succeeding instruction that needs it.



GURE 10.16 FPU pipeline stages

BYPASS1

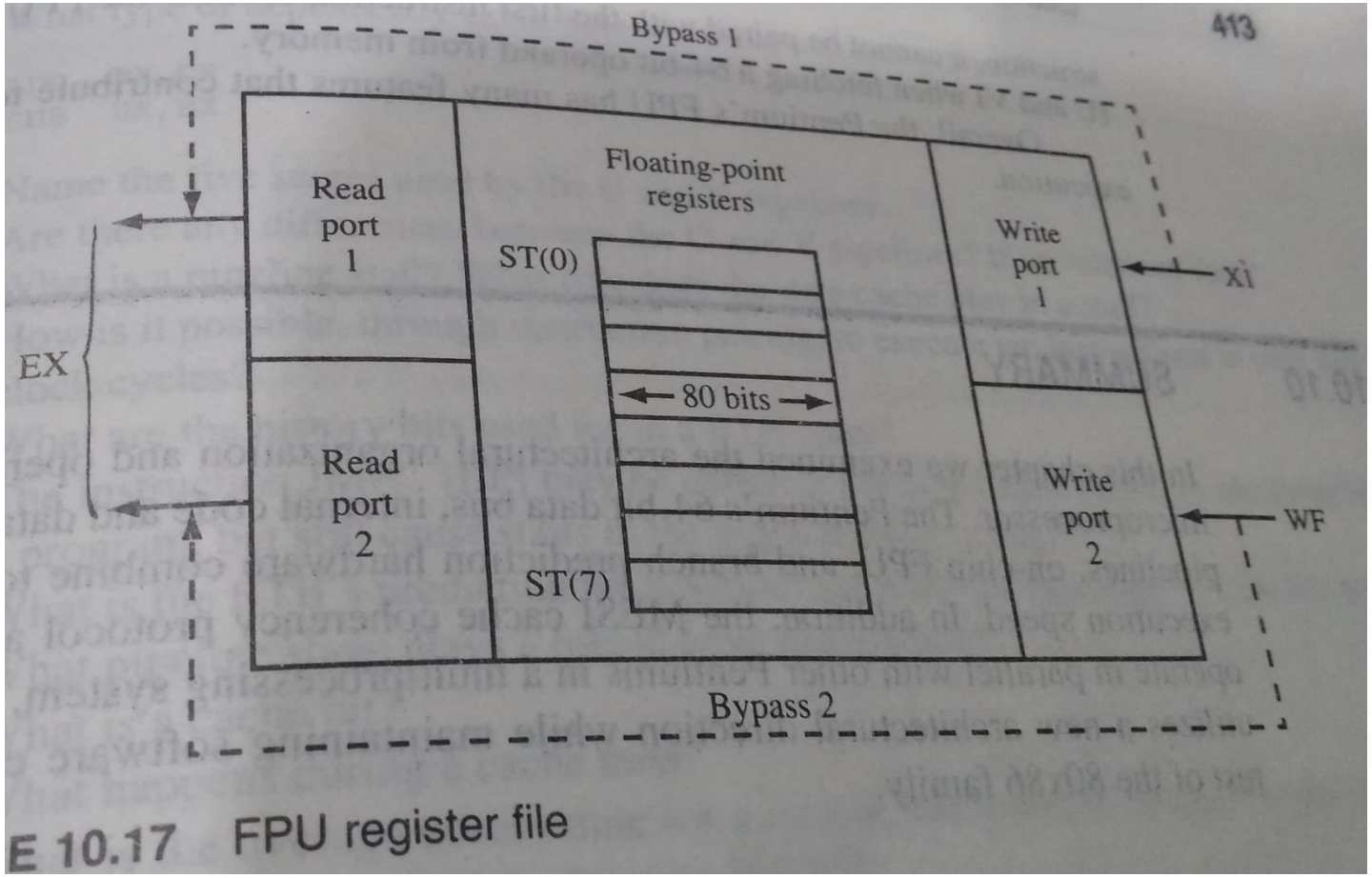
- **Bypass connects the output of X1 stage to the input of EX stage**

  e.g - FLD ST

  FMUL ST

- The above instructions use ST as an operand.
- With byapss1, the data from FLD is made available as an input operand to FMUL before it is written into the floating point register file.
- This prevents the pipeline from stalling and decreases number of clk cycles.
- No of clk cycles taken by 8087 to execute FMUL instruction are 130 wheras Pentium takes only 1 clk cycle to peform the same operation!

FPU Register File



E 10.17 FPU register file

BYPASS2

- **Bypass between WF stage and EX stage** where the result of an arithmetic instruction is made available to the next instruction.

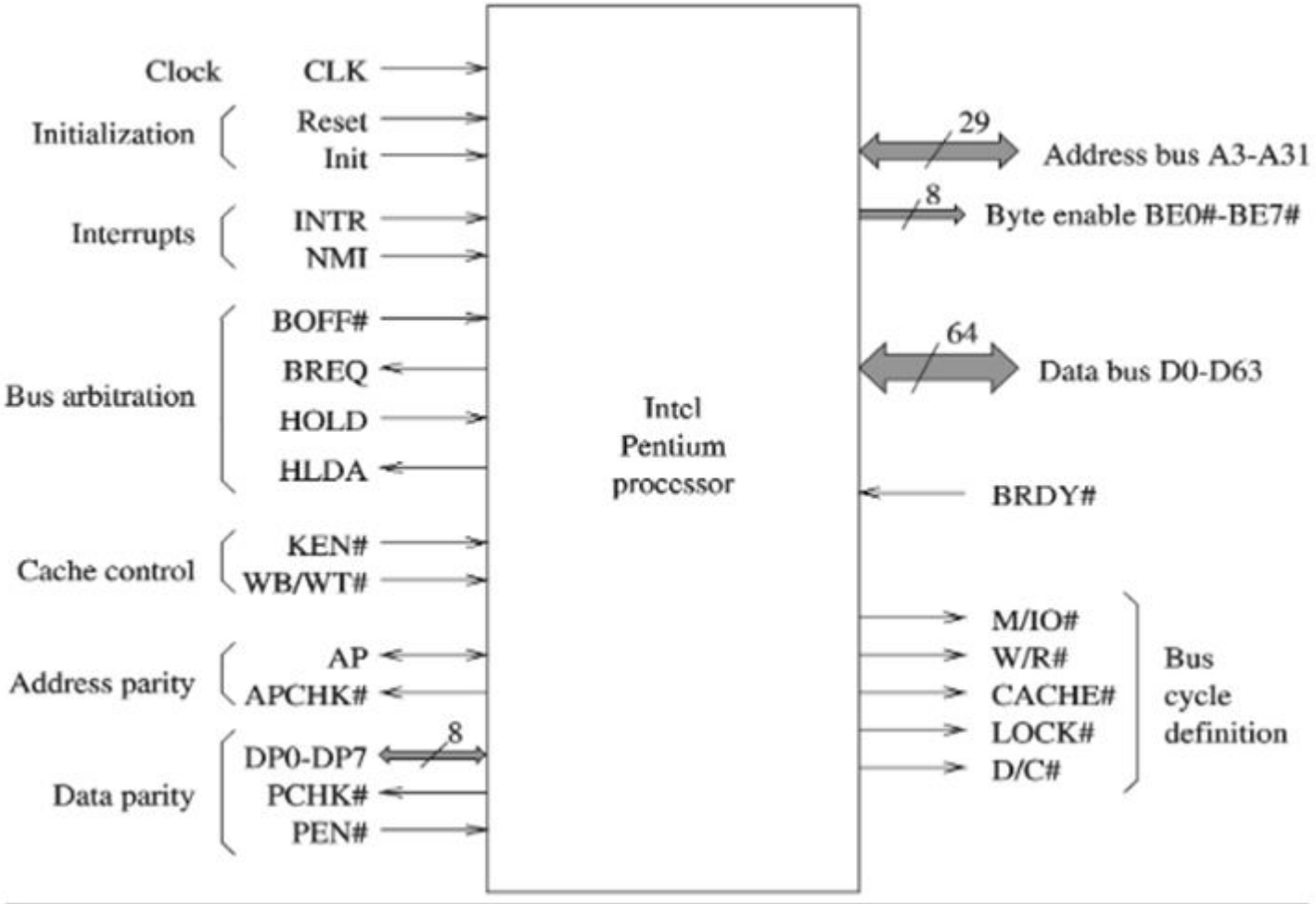
  e.g. – FADD NUM

  FMUL NUM1

- The first instruction does
  - Top of stack ← Top of stack + NUM
  - It does this in WF stage of the pipeline.
- The next instruction "FMUL NUM1" does
  - Top of stack ← Top of stack X NUM1

- *With bypass2, even before the result is written to top of the stack in register file, the result is directly forwarded to the next instruction "*FMUL NUM1*" in its EX stage.*

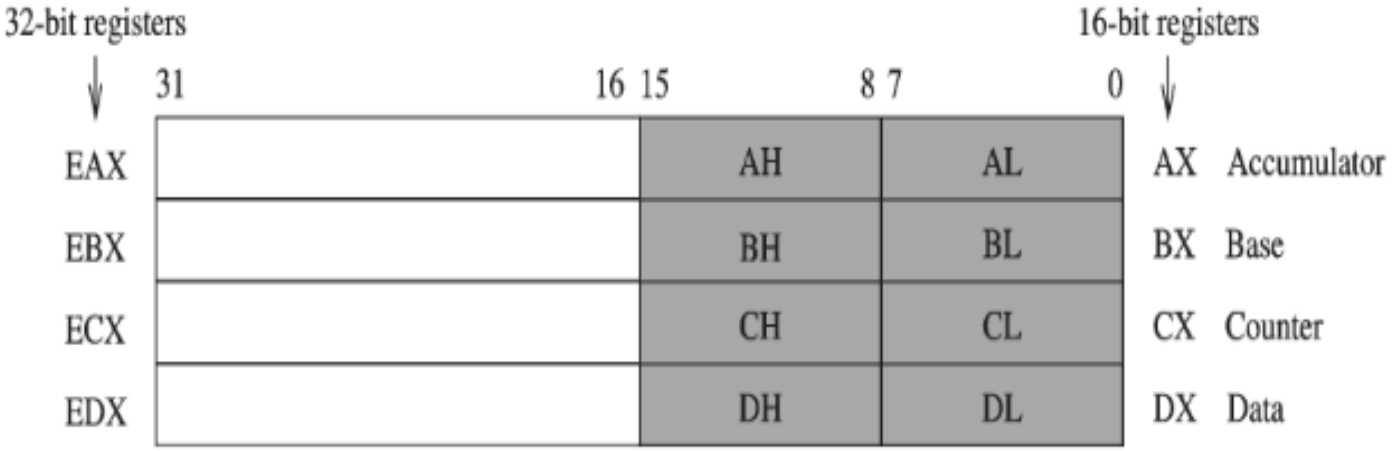
Pin Diagram


Clock
CLK
Initialization
Reset
Init
Interrupts
INTR
NMI
Bus arbitration
BOFF#
BREQ
HOLD
HLDA
Cache control
KEN#
WB/WT#
Address parity
AP
APCHK#
Data parity
8
DP0-DP7
PCHK#
PEN#
Intel
Pentium
processor
29
Address bus A3-A31
8
Byte enable BE0#-BE7#
64
Data bus D0-D63
BRDY#
M/IO#
W/R#
CACHE#
LOCK#
D/C#
Bus
cycle
definition

Pentium programmer's model

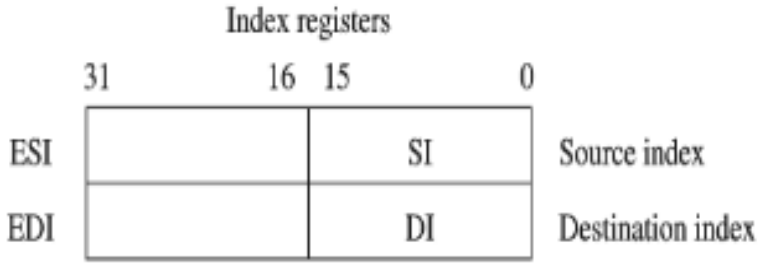
Pentium Registers

- Four 32-bit registers can be used as
  - Four 32-bit register (EAX, EBX, ECX, EDX)
  - Four 16-bit register (AX, BX, CX, DX)
  - Eight 8-bit register (AH, AL, BH, BL, CH, CL, DH, DL)
- Some registers have special use
  - ECX for count in loop instructions

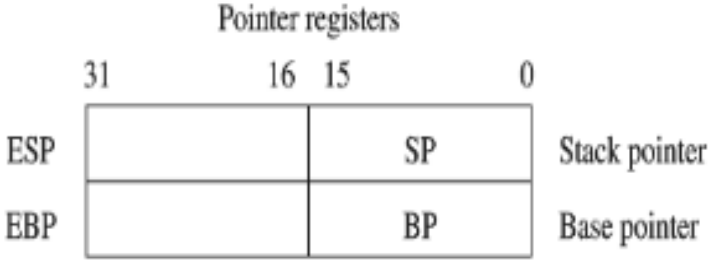


Continued…

- Two index registers
  - 16- or 32-bit registers
  - Used in string instructions
    - Source (SI) and destination (DI)
  - Can be used as general-purpose data registers

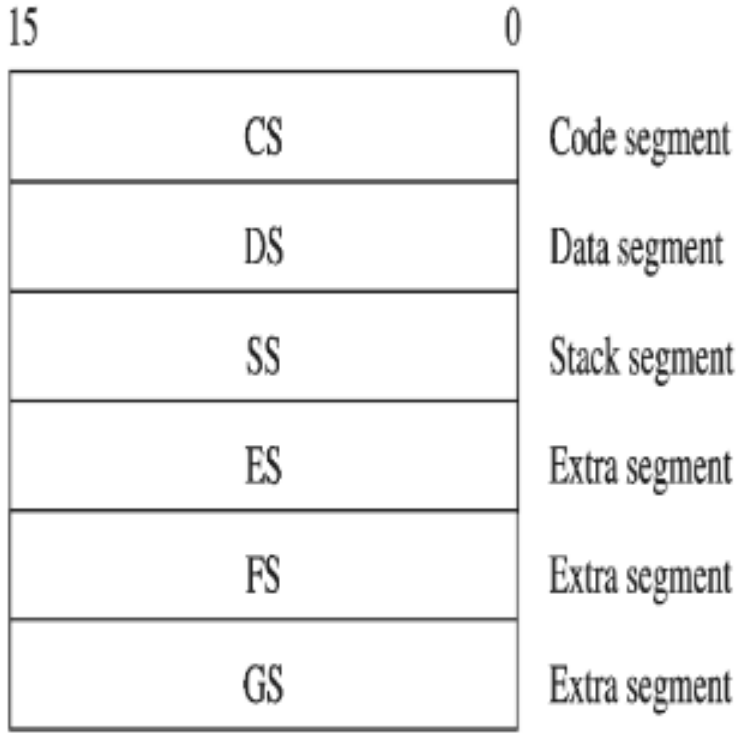


- Two pointer registers
  - 16- or 32-bit registers
  - Used exclusively to maintain the stack



Segment Registers

- Segment register
  - * Six 16-bit registers
  - * Support segmented memory architecture
  - * At any time, only six segments are accessible
  - * Segments contain distinct contents
    - » Code
    - » Data
    - » Stack



System Registers

To assist in initializing the processor and controlling system operations, the system architecture provides system flags in the EFLAGS register and several system registers:

- The system flags and IOPL field in the EFLAGS register control task and mode switching, interrupt handling, instruction tracing, and access rights.
- The control registers (CR0, CR2, CR3, and CR4) contain a variety of flags and data fields for controlling system-level operations.
- The debug registers allow the setting of breakpoints for use in debugging programs and systems software.
- The GDTR, LDTR, and IDTR registers contain the linear addresses and sizes (limits) of their respective tables.
- The task register contains the linear address and size of the TSS for the current task.

Memory Management Registers

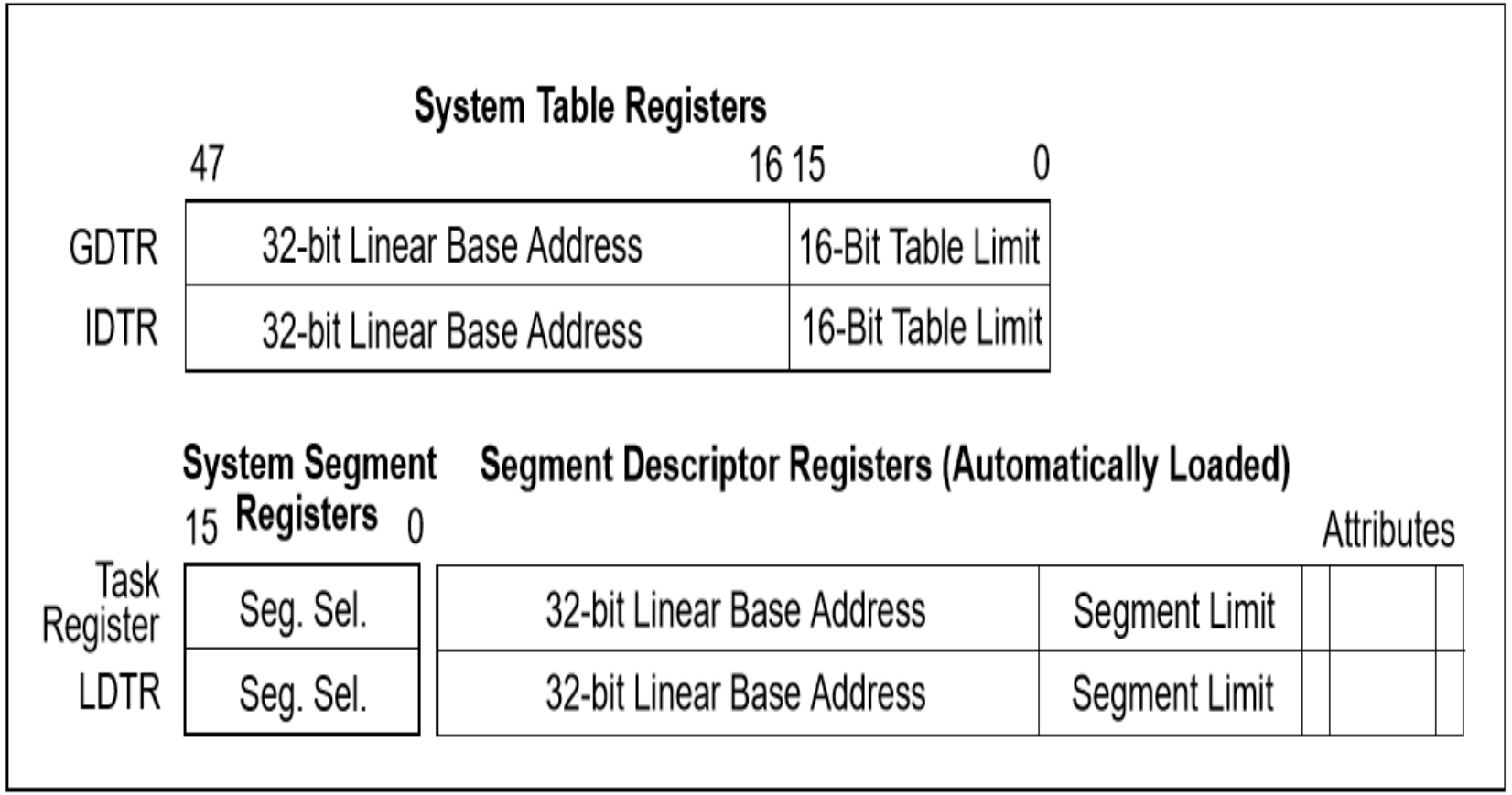
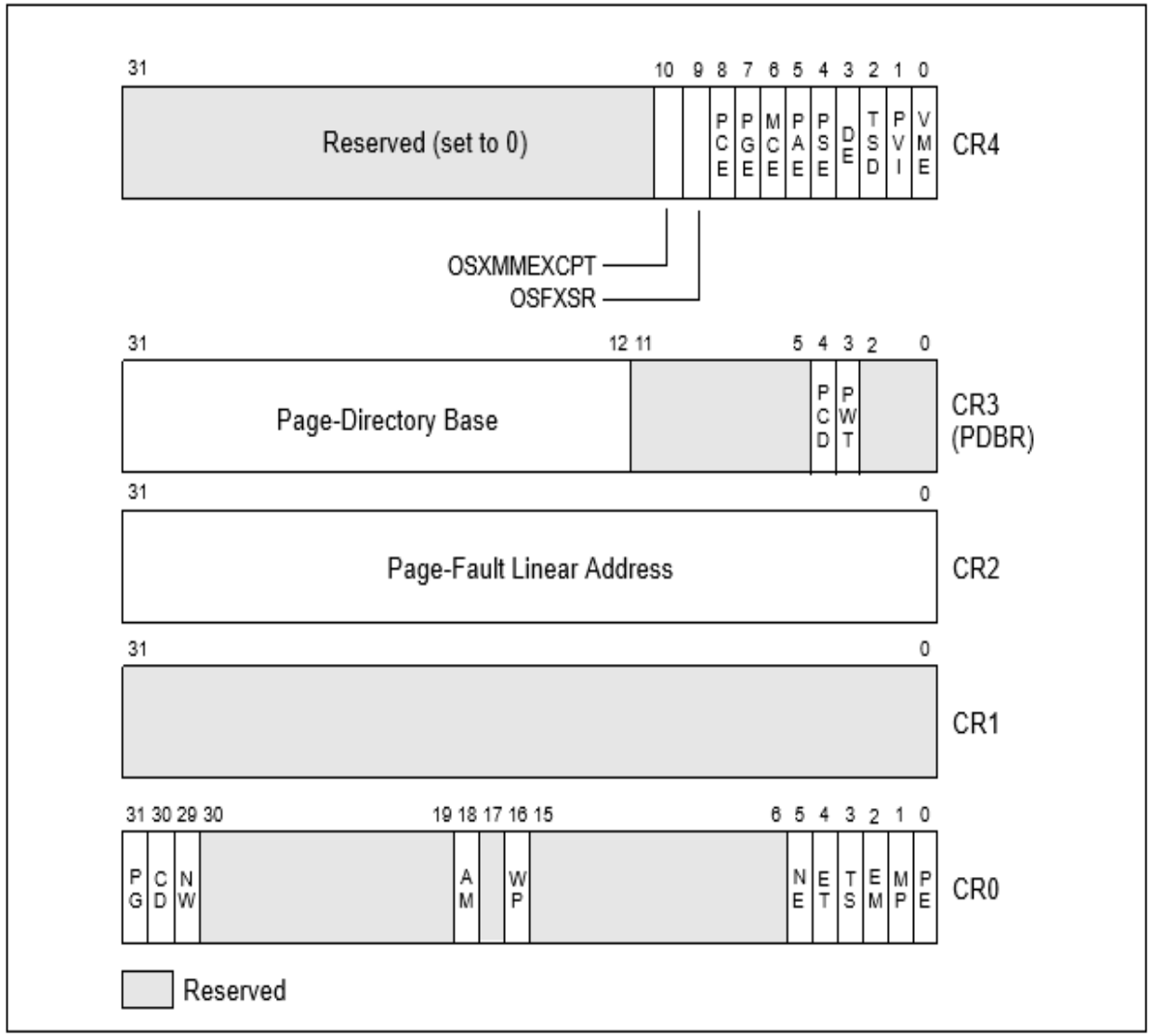


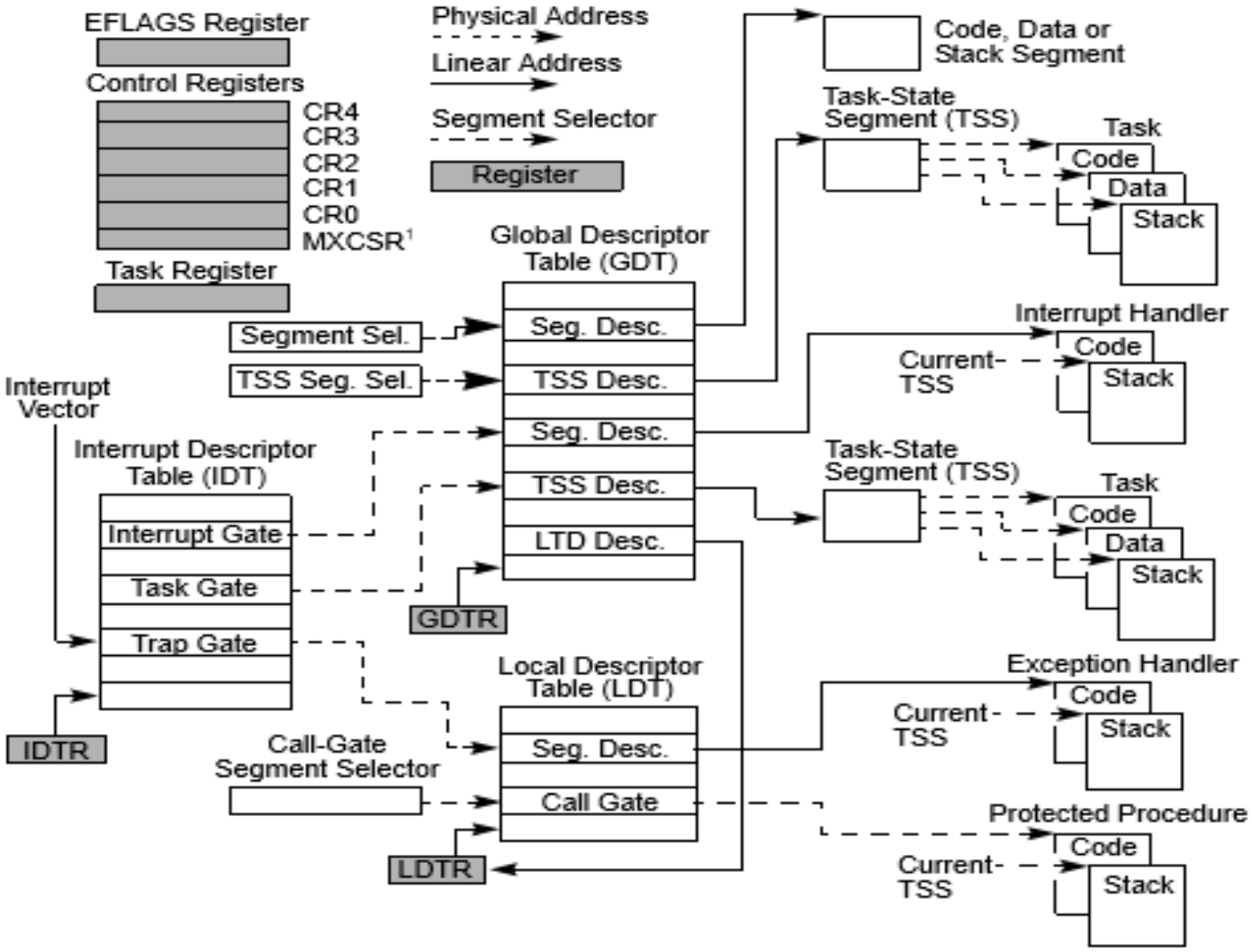
Figure 2-4. Memory Management Registers

Control Registers

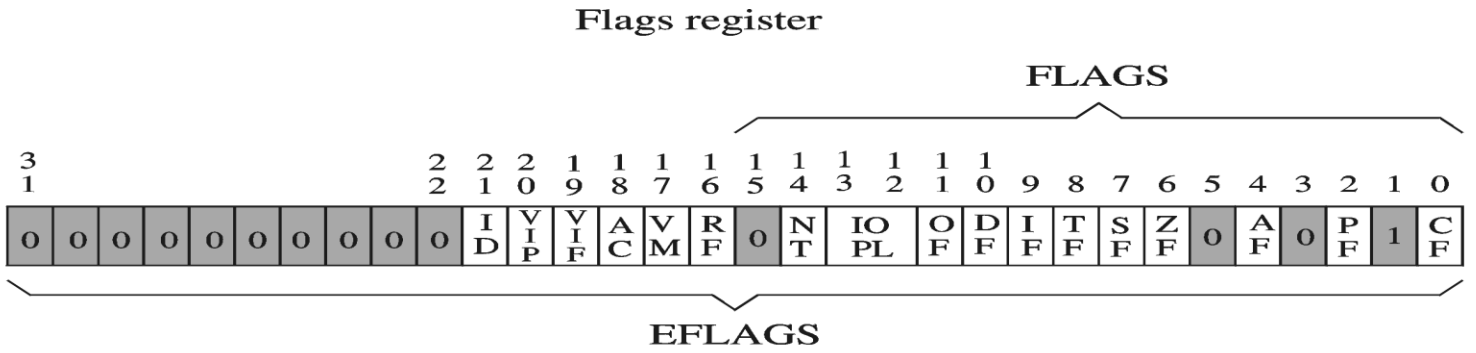
The control registers (CR0, CR1, CR2, CR3, and CR4) determine operating mode of the processor and the characteristics of the currently executing task



System Registers



EFLAGS Register



Status flags

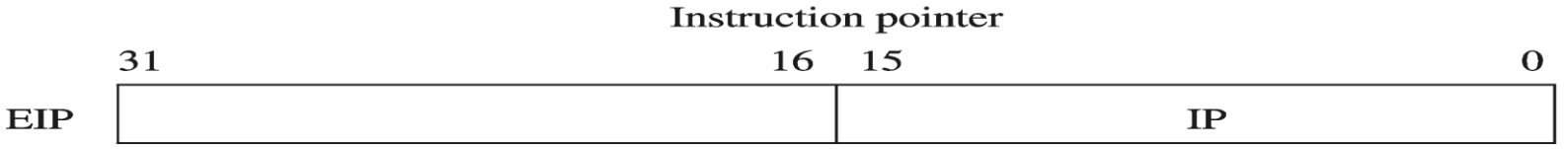
CF = Carry flag
PF = Parity flag
AF = Auxiliary carry flag
ZF = Zero flag
SF = Sign flag
OF = Overflow flag

Control flags

DF = Direction flag

System flags

TF = Trap flag
IF = Interrupt flag
IOPL = I/O privilege level
NT = Nested task
RF = Resume flag
VM = Virtual 8086 mode
AC = Alignment check
VIF = Virtual interrupt flag
VIP = Virtual interrupt pending
ID = ID flag



EFLAGS Register

TF Trap

- In single-step mode, the processor generates a debug exception after each instruction, which allows the execution state of a program to be inspected after each instruction.
- If an application program sets the TF flag using a POPF, POPFD, or IRET instruction,
- a debug exception is generated after the instruction that follows the POPF, POPFD, or IRET instruction.

IF Interrupt enable

- Controls the response of the processor to maskable hardware interrupt requests .
- The CPL,IOPL, and the state of the VME flag in control register CR4 determine whether the IF flag can be modified by the CLI, STI, POPF, POPFD, and IRET instructions.

EFLAGS Register

NT Nested task

Controls the chaining of interrupted and called tasks.

The flag can be explicitly set or cleared with the POPF/POPFD instructions;

RF Resume

Controls the processor's response to instruction-breakpoint conditions.

VM Virtual-8086 mode

Set to enable virtual-8086 mode; clear to return to protected mode.

AC Alignment check

Set this flag and the AM flag in the CR0 register to enable alignment checking of memory references

EFLAGS Register

IOPL I/O privilege level field

- Indicates the I/O privilege level (IOPL) of the currently running program or task.
- The CPLof the currently running program <= the IOPL to access the I/O address space.
- This field can only be modified by the POPF and IRET instructions when operating at a CPL of0

VIF Virtual Interrupt

- Contains a virtual image of the IF flag.
- This flag is used in conjunction with the VIP flag. The processor only recognizes the VIF flag when either the VME flag or the PVI flag in control register CR4 is set and the IOPL is less than 3.

EFLAGS Register

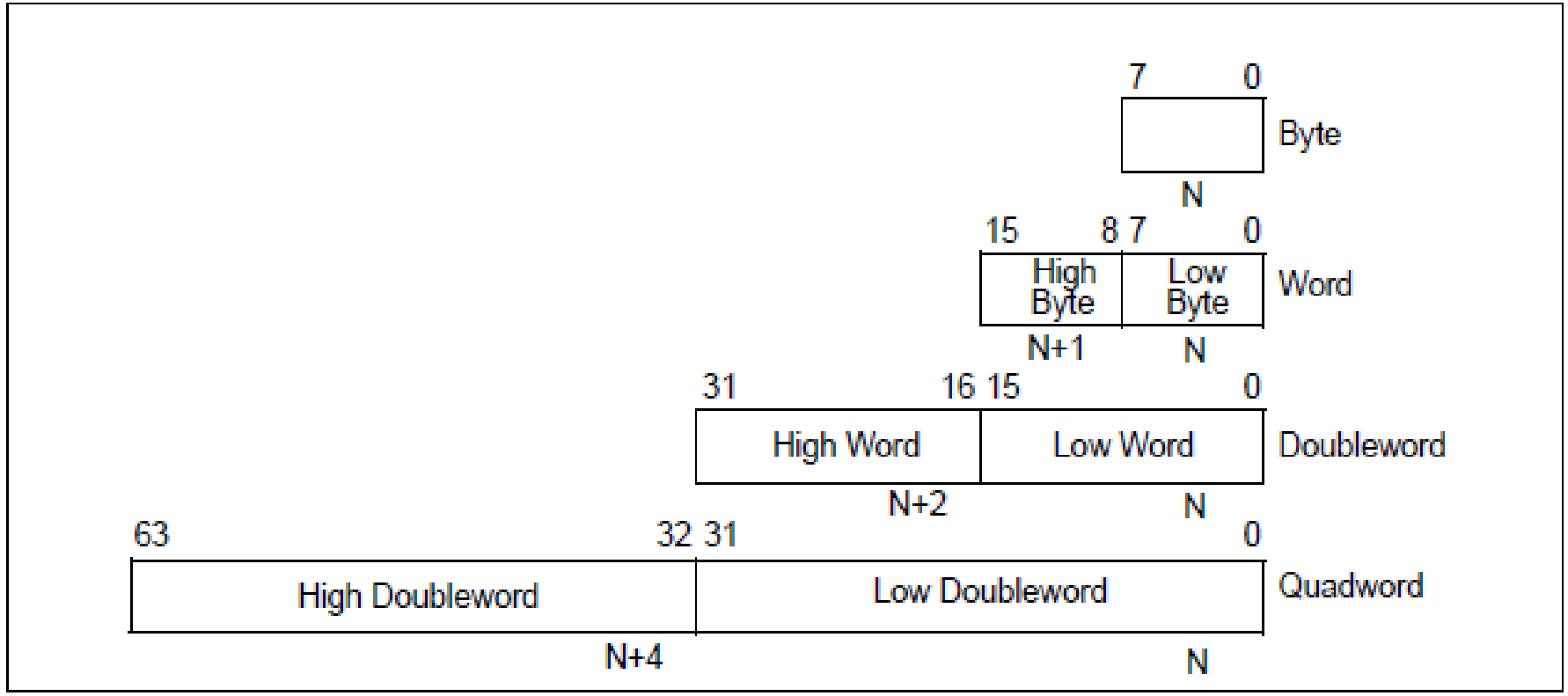
VIP Virtual interrupt pending

- Set by software to indicate that an interrupt is pending; cleared to indicate that no interrupt is pending. This flag is used in conjunction with the VIF flag.
- The processor reads this flag but never modifies it.
- The processor only recognizes the VIP flag when either the VME flag or the PVI flag in control register CR4 is set and the IOPL is less than 3.

ID Identification

- The ability of a program or procedure to set or clear this flag indicates support for the CPUID instruction.

Fundamental Data Types



Data Types

- **Integers**

  Integer values range

  from –128 to +127 for a byte integer

  from –32,768to +32,767 for a word integer

  from -2^{31} to $+2^{31}-1$ for a doubleword integer.

- Unsigned Integers(ordinals)

  from 0 to 255 for an unsigned byte integer,
  from 0 to 65,535 for an unsigned word integer,

  from 0 to $2^{32}-1$ for an unsigned doubleword integer

- BCD integer data types.

- **Floating-Point Data Types**

  The processor's floating-point instructions recognize a set of real, integer, and BCD integer data types.

Addressing Modes

Memory Operands

- Source and destination operands in memory are referenced by means of a segment selector and an offset.

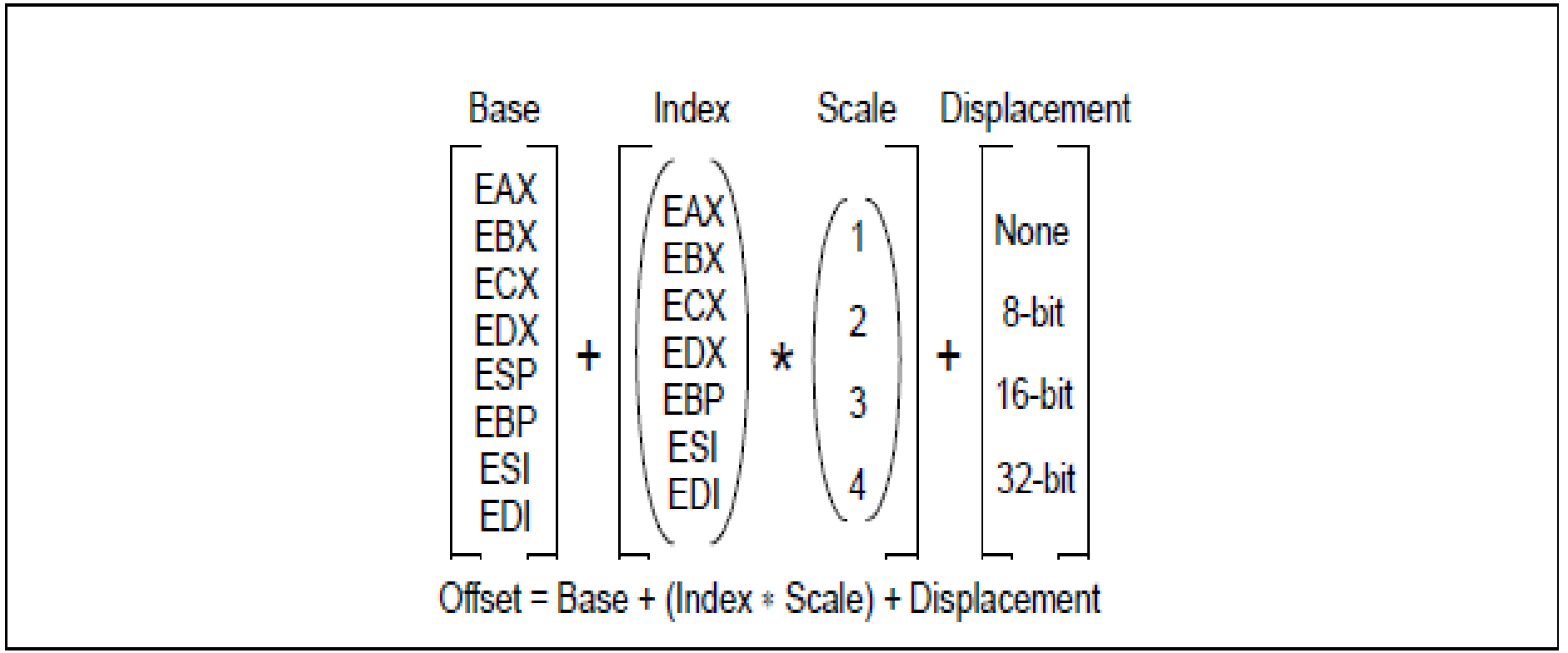
  e.g. MOV DS, BX

Specifying offset

The offset part of a memory address can be specified either directly as an static value (called a displacement) or through an address computation made up of one or more of the following components:

- **Displacement—An 8, 16, or 32 bit value.**
- **Base—The value in a general-purpose register.**
- **Index—The value in a general-purpose register.**
- **Scale factor—A value of 2, 4, or 8 that is multiplied by the index va**lue.

Offset (or Effective Address) Computation



- A scale factor(1,2,4,8) may be used only when an index also is used
- The ESP register cannot be used as an index register.
- When the ESP or EBP register is used as the base, the SS segment is the default segment.
- In all other cases, the DS segment is the default segment.

Addressing Modes

1. Register Operand Mode: The operand is located in one of the 8-, 16- or 32-bit general registers. e.g. MOV EAX,EBX

2. Immediate Operand Mode: The operand is included in the instruction as part of the opcode.
E.g. SHR PATTERN, 2

3. The remaining 9 modes provide a mechanism for specifying the effective address of an operand.

The effective address is calculated by using combinations of the following four address elements:

DISPLACEMENT,BASE,INDEX ,SCALE

EA =Base Reg +(Index Reg * Scaling) +Displacement

MIT-WPU
।। विश्वशान्तिर्ध्रुवं ध्रुवा ।।

- **Direct Mode**: The operand's offset is contained as part of the instruction as an 8-, 16- or 32-bit displacement.
  - EXAMPLE: MOV EAX,[1500H]
- **Register Indirect Mode**: A BASE register contains the address of the operand.
  - EXAMPLE: MOV ECX, [EDX]
- **Based Mode**: A BASE register's contents is added to a DISPLACEMENT to form the operands offset.
  - EXAMPLE: MOV ECX, [EAX+24]
- **Index Mode**: An INDEX register's contents is added to a DISPLACEMENT to form the operands offset.
  - EXAMPLE: ADD EAX, TABLE[ESI]
- **Scaled Index Mode**: An INDEX register's contents is multiplied by a scaling factor which is added to a DISPLACEMENT to form the operands offset.
  - EXAMPLE: IMUL EBX, TABLE[ESI*4]

MIT-WPU
।। विश्वशान्तिर्ध्रुवं ध्रुवा ।।

- **Based Index Mode**: The contents of a BASE register is added to the contents of an INDEX register to form the effective address of an operand.
  - EXAMPLE: MOV EAX, [ESI] [EBX]

- **Based Scaled Index Mode**: The contents of an INDEX register is multiplied by a SCALING factor and the result is added to the contents of a BASE register to obtain the operands offset.
  - EXAMPLE: MOV ECX, [EDX*8] [EAX]

- **Based Index Mode with Displacement**: The contents of an INDEX Register and a BASE register's contents and a DISPLACEMENT are all summed together to form the operand offset.
  - EXAMPLE: ADD EDX, [ESI] [EBP+00FFFFF0H]

- **Based Scaled Index Mode with Displacement**: The contents of an INDEX register are multiplied by a SCALING factor, the result is added to the contents of a BASE register and a DISPLACEMENT to form the operand's offset.
  - EXAMPLE: MOV EAX,[EBX] [EDI*4] +24

Addressing Modes

1.	Immediate Addressing:	Mov EAX,1000H
2.	Register Addressing :	Mov EAX,EBX
3.	Direct Addressing :	Mov EAX,[1234H]
4.	Register Indirect Addressing	Mov EDX,[ESI]
5.	Base Addressing:	Mov EAX,[EBX+4]
6.	Based Indexed Addressing:	Mov EAX,[EBX][ECX * 4]
7.	String Addressing:MOVSB	
8.	Port addressing : IN AL,80H	

Base + Displacement

(Index * Scale) + Displacement

Base + Index + Displacement

Base + (Index * Scale) + Displacement

MIT-WPU

।। विश्वशान्तिर्ध्रुवं ध्रुवा ।।

Pentium Instruction Set

Data Transfer Instructions

• MOV	Move
• XCHG	Exchange
• PUSH/PUSHF	Push onto stack
• POP /POPF	Pop off of stack
• SAHF	store AH into flags
• IN	Read from a port
• OUT	Write to a port
• LDS	Load far pointer using DS
• LES	Load far pointer using ES
• LEA	Load effective address
• XLAT	Exchange byte/word

Data Transfer Instructions

- Additional Pentium instructions:

• **PUSHA**	Push ALL registers
• **POPA**	Pop ALL registers
• INS	Input string from port
• OUTS	Output string from port
• **LFS**	Load far pointer using FS
• **LGS**	Load far pointer using GS
• **MOVSX**	Move and sign extend
• **MOVZX**	Move and zero extend
• **PUSHD/PUSHAD/PUSHFD**	Push DOUBLE registers onto stack
• **POPD/POPAD /POPF**D	Pop DOUBLE registers from stack
• BSWAP	Byte swap
• MOV	Move to/from control register

Stack Manipulation Instructions

- PUSH (Push) decrements the stack pointer (ESP), then transfers the source operand to the top of stack indicated by ESP
- PUSH is often used to place parameters on the stack before calling a procedure.
- The PUSH instruction operates on memory operands, immediate operands, and register .

- PUSHA (Push All Registers) saves the contents of the eight general registers on the stack..
- The processor pushes the general registers on the stack in the following order:
- EAX, ECX, EDX, EBX, the initial value of ESP before EAX was pushed, EBP, ESI, and EDI.

- LDS dest,src (Load pointer using DS)
  - Load two 16 bit registers from a 4 byte block of memory.
  - First 2 bytes are copied into dest register.
  - Second 2 bytes are copied into DS register.

e.g. DS=0100,SI=0020

Command:LDS DI,[SI]

Physical addr=01020h

Result:DI = F0 30

DS= 90 80

Address	Memory
01020	30
01021	F0
01022	80
01023	90
01024	

LEA

- The lea(Load Effective Address) instruction is another instruction used to prepare pointer values.
- lea dest, source

e.g. LEA esi,msg

lea reg16, mem

lea reg32, mem

- MOVSX dest,src :
  - Move with sign extended
  - Convert 8/16 bit nos into 16/32 bit sign extended nos.

E.g. Given AL=36h,BX= C3EEh

Command:MOVSX AX,AL

Result: AX= 0036h

Command:MOVSX EBX,BX

Result: EBX=FFFFC3EEh

- MOVZX dest,src :
  - Move with zero extended
  - Convert 8/16 bit nos into 16/32 bit sign extended nos.

E.g. Given AL=36h,BX= C3EEh

Command:MOVSX AX,AL

Result: AX= 0036h

Command:MOVSX EBX,BX

Result: EBX=0000C3EEh

Binary Arithmetic Instructions

- ADD Integer add
- ADC Add with carry
- SUB Subtract
- SBB Subtract with borrow
- IMUL Signed multiply
- MUL Unsigned multiply
- IDIV Signed divide
- DIV Unsigned divide
- INC Increment
- DEC Decrement
- NEG Negate
- CMP Compare

The 80486 has 6-new instructions more to that of the 80386 and these are

BSWAP	⇒ Byte Swap.
CMPXCHG	⇒ Compare and Exchange.
XADD	⇒ Exchange and ADD.

Byte Swap is especially suitable for downloading of mainframe data on an 80486 system. Little-endian data is stored with the most significant bits in the highest-addressed byte. Big-endian data is stored with the most significant bits at lowest addressed byte. The instruction swap reverses the order of the 4-bytes of the 32-bit word thus allowing data conversion from Little-endian format to Big-endian format and vice-versa. Other 2-instructions CMPXCHG and XADD make multi-processing and multitasking capabilities more efficient.

Binary Arithmetic Instructions

- Additional Pentium instructions:
  - CDQ convert double word to quad word
  - CWDE convert word to double word

  - CMPXCHG compare and exchange
  - XADD exchange and add

  - CMPXCHG8B compare and exchange 8bytes

- CWDE (Convert word to Double word)
  - Extends sign of no. stored in AX through upper 16 bits of EAX.i.e. 32 bit signed no. is stored in EAX.
  - Useful for IMUL,IDIV

  E.g. AX= 4000h

  CWDE

  Result: EAX=00004000h

- CDQ(Convert Double word to Quad word)
  - Extends sign of no. stored in EAX through EDX i.e. 64 bit signed no. is stored in EDX:EAX.
  - Useful for IMUL,IDIV

  E.g.EAX= C8000000h

  CDQ

  Result: EDX=FFFFFFFF,EAX= C8000000h

Bit manipulation Instructions

- AND and
- OR or
- XOR Exclusive or
- NOT Not
- SAR Shift arithmetic right
- SHR Shift logical right
- SAL/SHL Shift arithmetic left/Shift logical left
- SHRD Shift right double
- SHLD Shift left double
- ROR Rotate right
- ROL Rotate left
- RCR Rotate through carry right
- RCL Rotate through carry left
- TEST test byte/word

Bit manipulation Instructions

- Additional Pentium instructions:
  - BSF Bit scan forward
  - BSR Bit scan reverse
  - BT Bit test
  - BTC Bit test and complement
  - BTR Bit test and reset
  - BTS Bit test and set
  - SHLD shift left double precision
  - SHRD shift right double precision
  - SETcc set byte on condition

- **BT/BTC/BTS/BTR dest,src (Bit test and complement/set/reset)**
  - Determines the value of specific bit in 16/32 bit dest.
  - Bit to be tested is indicated by src.
  - The bits are numbered from 0 to 31.
  - The state of the bit that is tested is copied into carry flag.
  - Useful in control applications.

E.g. AX=5555h,Carry=0

Command: BT AX,10
Result: Carry=1

Command: BTC AX,15
Result: Ax=D555,Carry=0

Command: BTS AX,1
Result: Ax=D557,Carry=0

Command: BTR AX,0
Result: Ax=D556,Carry=1

- BSR/BSF dest,src (Bit scan reverse/forward)
  - Scans src for first bit that equals 1(starting with LSB for BSF, starting with MSB for BSR)
  - The bit position of first 1 found is saved in dest.
  - Useful in edge detection in Image Processing.

E.g. AX=2000h,EBX=4CE00000h

0	1	0	0	1	1	0	0	1	1	1	0	0	0	.	.	0	0	0	0

Command: BSR EAX,EBX

Result EAX=1Eh(Bit 30)

0	1	0	0	1	1	0	0	1	1	1	0	0	0	.	.	0	0	0	0

Command: BSF EAX,EBX

Result EAX=15h(Bit 21)

Control Transfer Instructions

- JMP Jump
- JE/JZ Jump if equal/Jump if zero
- JA/JNBE Jump if above/Jump if not below or equal
- JL/JNGE Jump if less/Jump if not greater or equal
- JC Jump if carry
- JO Jump if overflow
- JS Jump if sign (negative)
- RET Return
- IRET Return from interrupt
- INT Software interrupt
- INTO Interrupt on overflow
- LOOPZ/LOOPE Loop with ECX and zero/Loop with ECX and equal

String Instructions

• MOVS/MOVSB	Move string/Move byte string
• CMPS/CMPSB	Compare string
• SCAS/SCASB	Scan string/Scan byte string
• LODS/LODSB	Load string/Load byte string
• STOS/STOSB	Store string/Store byte string
• REP	Repeat while ECX not zero
• REPE/REPZ	Repeat while equal/Repeat while zero
• REPNE/REPNZ	Repeat while not equal/Repeat while not zero

Flag Control Instructions

- STC Set carry flag
- CLC Clear the carry flag
- CMC Complement the carry flag
- CLD Clear the direction flag
- STD Set direction flag
- LAHF Load flags into AH register
- SAHF Store AH register into flags
- STI Set interrupt flag
- CLI Clear the interrupt flag

Additional Pentium instructions:

- BOUND Detect value out of range
- ENTER High-level procedure entry
- LEAVE High-level procedure exit

- IRETD interrupt return
- JECXZ jump if ECX is zero

Protected Mode Instructions

- LGDT — Load global descriptor table (GDT) register
- SGDT — Store global descriptor table (GDT) register
- LLDT — Load local descriptor table (LDT) register
- SLDT — Store local descriptor table (LDT) register
- LTR — Load task register
- STR — Store task register
- LIDT — Load interrupt descriptor table (IDT) register
- SIDT — Store interrupt descriptor table (IDT) register
- MOV — Load and store control registers
- MOV — Load and store debug registers

Protected Mode Instructions

- LMSW Load machine status word
- SMSW Store machine status word
- CLTS Clear the task-switched flag
- ARPL Adjust requested privilege level
- LAR Load access rights
- LSL Load segment limit
- VERR Verify segment for reading
- VERW Verify segment for writing

- Additional Pentium instructions:
  - INVD — Invalidate cache, no writeback
  - WBINVD — Invalidate cache, with writeback
  - INVLPG — Invalidate TLB Entry
  - LOCK (prefix) — Lock Bus
  - HLT — Halt processor

  - RSM — Return from system management mode (SSM)
  - RDMSR — Read model-specific register
  - WRMSR — Write model-specific register
  - RDPMC — Read performance monitoring counters
  - RDTSC — Read time stamp counter
  - CPUID — Processor Identification