Pentium Processor

1. **Features of Pentium:**

- Introduced in 1993 with clock frequency ranging from 60 to 66 MHz
- 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.
- The primary changes in Pentium Processor were:
- Superscalar Architecture
- Dynamic Branch Prediction
- Pipelined Floating-Point Unit
- Separate 8K Code and Data Caches
- Writeback MESI Protocol in the Data Cache
- 64-Bit Data Bus
- Bus Cycle Pipelining

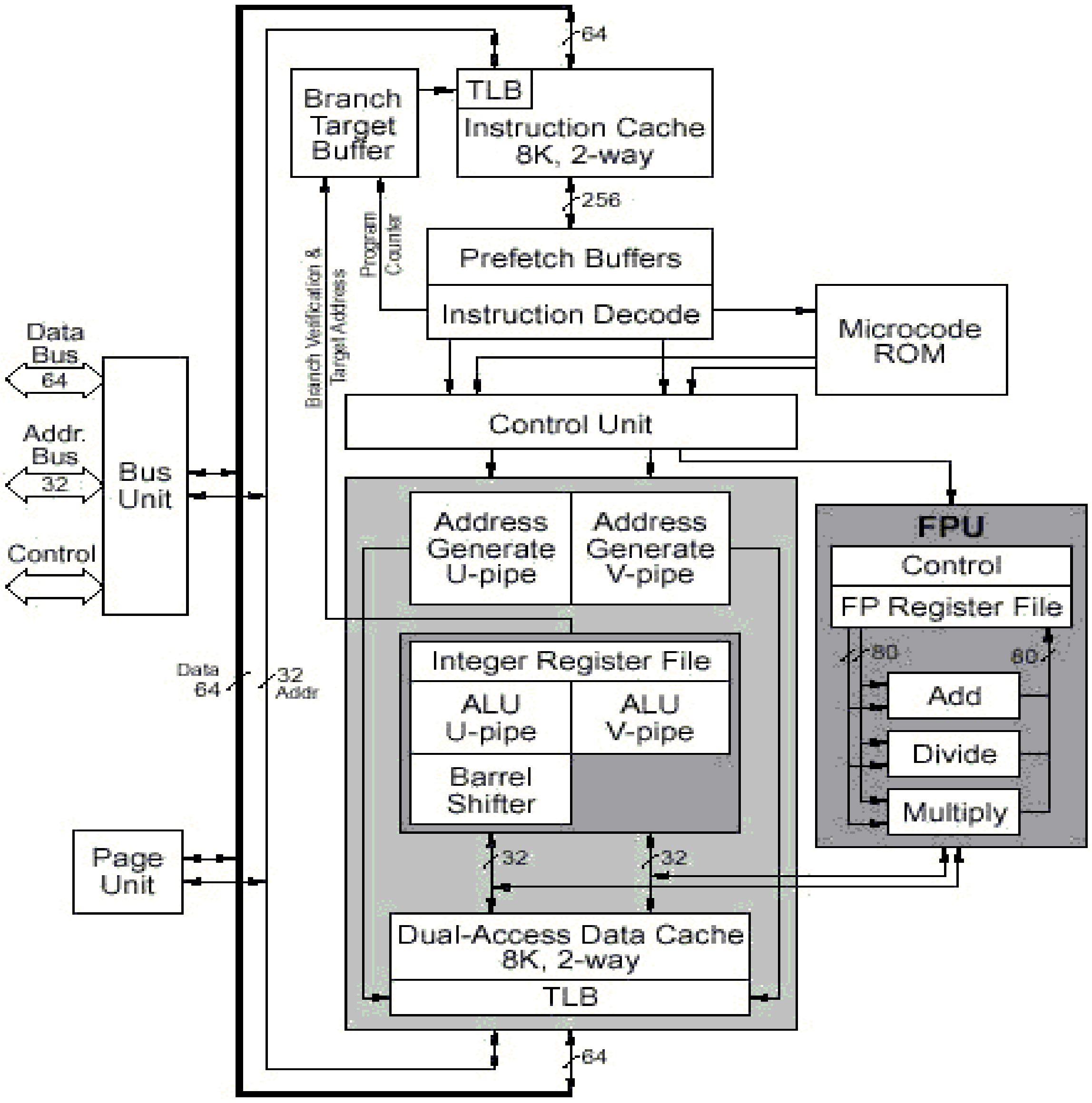
2. **Pentium Architecture:**

University Question: Draw and explain Pentium Processor architecture. Highlight architectural features

- 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
- **Prefetch Buffers:**
- Four prefetch buffers within the processor works as two independent pairs.
- 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.
- Prefetch buffer sends a pair of instructions to instruction decoder
- **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 :**
- This unit 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-pipeline has a barrel shifter
- The two ALUs perform the arithmetic and logical operations specified by their instructions in their respective pipeline
- **Address Generators :**
- Two address generators (one for each pipeline) form the address specified by the instructions in their respective pipeline.
- They are equivalent to segmentation unit.
- **Paging Unit :**
- If enabled, it translates linear address (from address generator) to physical address
- It can handle two linear addresses at the same time to support both pipelines with one TLB per cache
- **Floating Point Unit:**
- It can accept upto two floating point operations per clock when one of the instruction is an exchange instruction
- Three types of floating point operations can operate simultaneously within FPU: addition, division and multiplication.
- **Data Cache:**

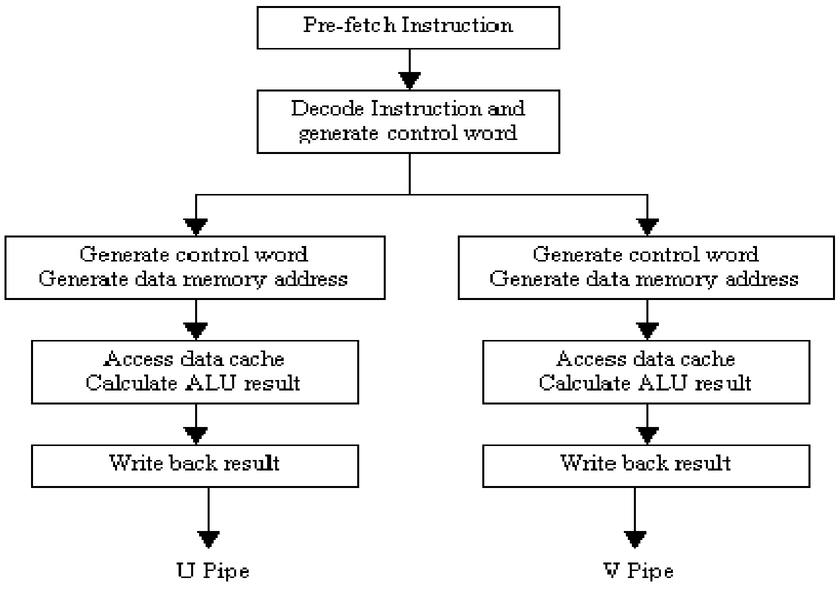
- It is an 8KB write-back , two way set associative cache with line size of 32 bytes
- **Bus Unit:**
- **Address Drivers and Receivers:**
- Push address onto the processor's local address bus (A31:A3 and BE7:BE0)
- **Data Bus Transceivers:**
- gate data into the processor 's local data bus
- **Bus Control Logic:**
- controls whether a standard or burst bus cycle is to be run
- **Branch Target Buffer:** supplies jump target prefetch addresses to the code cache



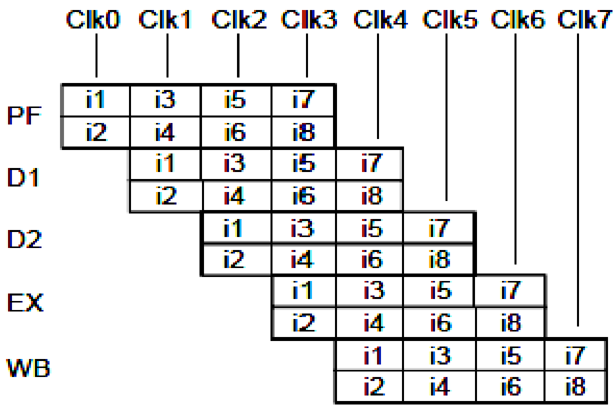
3. **Superscalar Operation:**

UQ: Explain with block diagram how superscalar operation is carried out in Pentium Processor

- The prefetcher sends an address to code cache and if present, a line of 32 bytes is send to one of the prefetch buffers
- The prefetch buffer transfers instructions to decode unit
- Initially it checks if the instructions can be paired.
- If paired, one goes to 'u' and other goes to 'v' pipeline as long as no dependencies exist between them.
- Pair of instructions enter and exit each stage of pipeline in unison.


Pre-fetch Instruction
Decode Instruction and generate control word
Generate control word Generate data memory address
Generate control word Generate data memory address
Access data cache Calculate ALU result
Access data cache Calculate ALU result
Write back result
Write back result
U Pipe
V Pipe

- Pentium uses a five stage execution pipeline as shown:

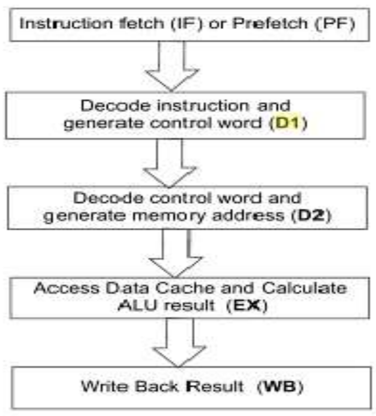


	Clk0	Clk1	Clk2	Clk3	Clk4	Clk5	Clk6	Clk7
PF	i1	i3	i5	i7				
	i2	i4	i6	i8				
D1		i1	i3	i5	i7			
		i2	i4	i6	i8			
D2			i1	i3	i5	i7		
			i2	i4	i6	i8		
EX				i1	i3	i5	i7	
				i2	i4	i6	i8	
WB					i1	i3	i5	i7
					i2	i4	i6	i8

4. **Integer Pipeline:**

UQ: Explain in brief integer instruction pipeline stages of Pentium

- The pipelines are called "u" and "v" pipes.
- The u-pipe can execute any instruction, while the v-pipe can execute "simple" instructions as defined in the "Instruction Pairing Rules".
- When instructions are paired, the instruction issued to the v-pipe is always the next sequential instruction after the one issued to u-pipe.
- The integer pipeline stages are as follows:


Instruction fetch (IF) or Prefetch (PF)
Decode instruction and generate control word (D1)
Decode control word and generate memory address (D2)
Access Data Cache and Calculate ALU result (EX)
Write Back Result (WB)

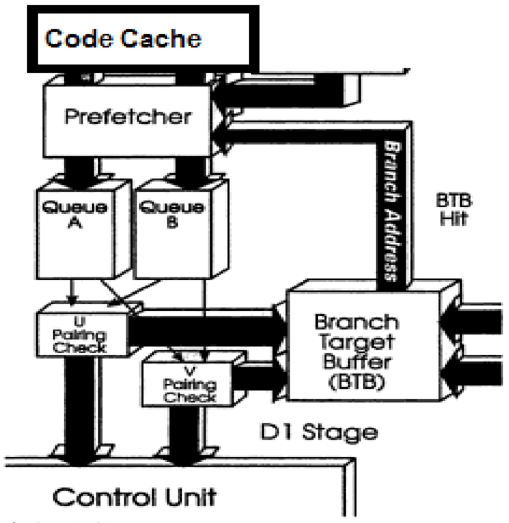
Prefetch (PF):

- Instructions are prefetched from the on-chip instruction cache

Decode1 (D1):

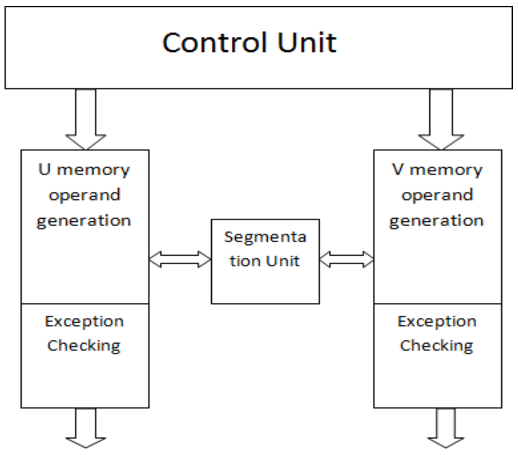
- Two parallel decoders attempt to decode and issue the next two sequential instructions

- It decodes the instruction to generate a control word
- A single control word causes direct execution of an instruction
- Complex instructions require microcoded control sequencing



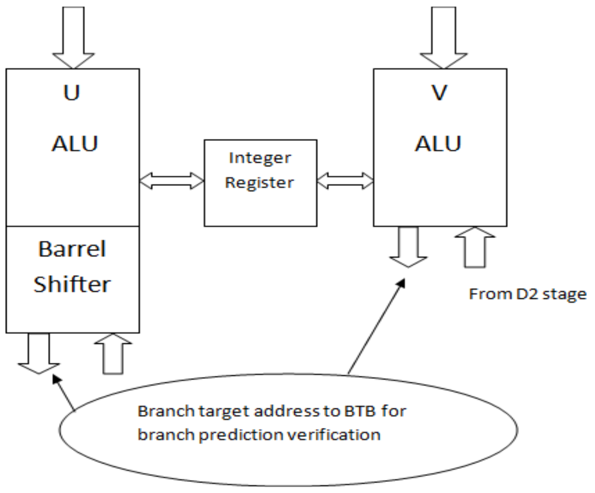
Decode2 (D2):

- Decodes the control word
- Address of memory resident operands are calculated



Execute (EX):

- The instruction is executed in ALU
- Data cache is accessed at this stage
- For both ALU and data cache access requires more than one clock.


U
ALU
Barrel Shifter
Integer Register
V
ALU
From D2 stage
Branch target address to BTB for branch prediction verification

Writeback(WB):

- The CPU stores the result and updates the flags

5. Integer Instruction Pairing Rules:

UQ: Explain the instruction pairing rules for Pentium

- To issue two instructions simultaneously they must satisfy the following conditions:
- Both instructions in the pair must be "simple". Simple Instructions:
- They are entirely hardwired
- They do not require any microcode control
- Executes in one clock cycle

-Exception: ALU mem,reg and ALU reg,mem are 3 and 2 clock operations respectively

- The following integer instructions are considered simple and may be paired:

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/jc near
11. nop	12. test reg, reg/mem
13. test acc, imm	

- There must be no read-after-write(RAW) or write-after-write register(WAW) dependencies

RAW:
i1. R2 € R1 + R3 i2. R4 € R2 + R3
WAW:
i1. R2 € R4 + R7 i2. R2 € R1 + R3

- The two instructions should not have immediate values.
- Instruction with prefixes (lock,repne) can only occur in the u-pipe

6. Instruction Issue Algorithm:

UQ: List the steps in instruction issue algorithm

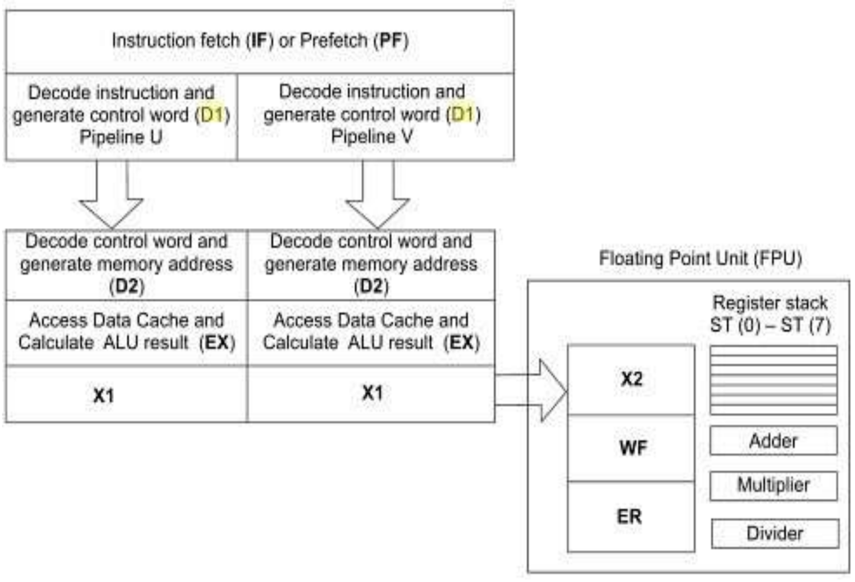
- Decode the two consecutive instructions I1 and I2

- If the following are all true:
- I1 and I2 are simple instructions
- I1 is not a jump instruction
- Destination of I1 is not a source of I2
- Destination of I1 is not a destination of I2
- Then issue I1 to u pipeline and I2 to v pipeline
- Else issue I1 to u pipeline

7. Floating Point Pipeline:

UQ: Explain the floating point pipeline stages.

- The floating point pipeline has 8 stages as follows:


Instruction fetch (IF) or Prefetch (PF)
Decode instruction and generate control word (D1) Pipeline U
Decode instruction and generate control word (D1) Pipeline V
Decode control word and generate memory address (D2)
Decode control word and generate memory address (D2)
Access Data Cache and Calculate ALU result (EX)
Access Data Cache and Calculate ALU result (EX)
X1
X1
Floating Point Unit (FPU)
Register stack ST (0) – ST (7)
X2
WF
ER
Adder
Multiplier
Divider

Prefetch(PF) :

- Instructions are prefetched from the on-chip instruction cache

Instruction Decode(D1):

- Two parallel decoders attempt to decode and issue the next two sequential instructions
- It decodes the instruction to generate a control word
- A single control word causes direct execution of an instruction
- Complex instructions require microcoded control sequencing

Address Generate (D2):

- Decodes the control word
- Address of memory resident operands are calculated

Memory and Register Read (Execution Stage) (EX):

- Register read, memory read or memory write performed as required by the instruction to access an operand.

Floating Point Execution Stage 1(X1):

- Information from register or memory is written into FP register.
- Data is converted to floating point format before being loaded into the floating point unit

Floating Point Execution Stage 2(X2):

- Floating point operation performed within floating point unit.

Write FP Result (WF):

- Floating point results are rounded and the result is written to the target floating point register.

Error Reporting(ER)

- If an error is detected, an error reporting stage is entered where the error is reported and FPU status word is updated.

8. Instruction Issue for Floating Point Unit:

The rules of how floating-point (FP) instructions get issued on the Pentium processor are :

- FP instructions do not get paired with integer instructions.
- When a pair of FP instructions is issued to the FPU, only the FXCH instruction can be the second instruction of the pair.
- The first instruction of the pair must be one of a set F where F = [FLD,FADD, FSUB, FMUL, FDIV, FCOM, FUCOM, FTST, FABS, FCHS].
- FP instructions other than FXCH and instructions belonging to set F, always get issued singly to the FPU.
- FP instructions that are not directly followed by an FXCH instruction are issued singly to the FPU.

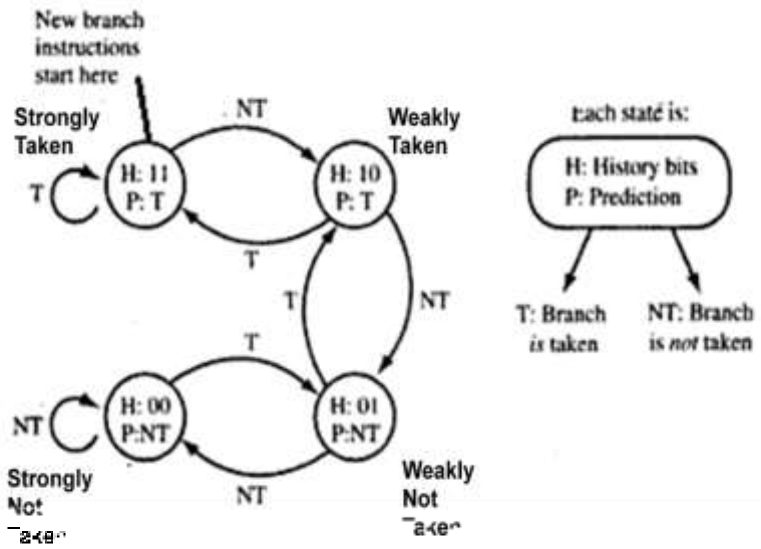
9. Branch Prediction Logic:

UQ : Explain how the flushing of pipeline can be minimized in Pentium Architecture

- 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.
- To avoid this problem, the Pentium uses a scheme called Dynamic Branch Prediction.
- 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.
- It results in a 3 cycle penalty if the branch is executed in the u-pipeline and 4 cycle penalty in v-pipeline.
- It is implemented using a 4-way set associative cache with 256 entries. This is referred to as the Branch Target Buffer(BTB).
- The directory entry for each line contains the following information:
- Valid Bit : Indicates whether or not the entry is in use
- History Bits: track how often the branch has been taken
- Source memory address that the branch instruction was fetched from (address of I3)
- If its directory entry is valid, the target address of the branch is stored in corresponding

data entry in BTB

- BTB is a look-aside cache that sits off to the side of D1 stages of two pipelines and monitors for branch instructions.
- The first time that a branch instruction enters either pipeline, the BTB uses its source memory address to perform a lookup in the cache.
- Since the instruction has not been seen before, this results in a BTB miss.
- It means the prediction logic has no history on instruction.
- It then predicts that the branch will not be taken and program flow is not altered.
- Even unconditional jumps will be predicted as not taken the first time that they are seen by BTB.
- When the instruction reaches the execution stage, the branch will be either taken or not taken.
- If taken, the next instruction to be executed should be the one fetched from branch target address.
- If not taken, the next instruction is the next sequential memory address.
- When the branch is taken for the first time, the execution unit provides feedback to the branch prediction logic.
- The branch target address is sent back and recorded in BTB.
- A directory entry is made containing the source memory address and history bits set as strongly taken



History Bits	Resulting Description	Prediction Made	If branch is taken	If branch is not taken
11	Strongly Taken	Branch Taken	Remains Strongly Taken	Downgrades to Weakly Taken
10	Weakly Taken	Branch Taken	Upgrades to Strongly Taken	Downgrades to Weakly Not Taken
01	Weakly Not Taken	Branch Not Taken	Upgrades to Weakly Taken	Downgrades to Strongly Not Taken
00	Strongly Not Taken	Branch Not Taken	Upgrades to Weakly Not Taken	Remains Strongly Not Taken

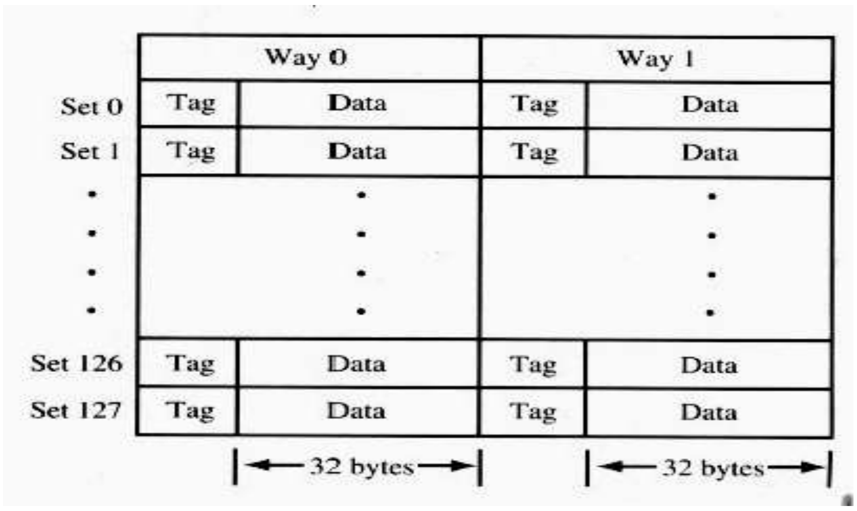
10. **Cache organization :**

UQ: Explain the cache organization of Pentium Processor

- The Pentium processor implements two internal caches for a total integrated cache size of 16

Kbytes: an 8 Kbyte data cache and a separate 8 Kbyte code cache.

- These caches are transparent to application software to maintain compatibility with previous Intel Architecture generations.
- 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.
- The caches have been designed for maximum flexibility and performance. The data cache is configurable as writeback or writethrough on a line-by-line basis.
- Memory areas can be defined as non-cacheable by software and external hardware. Cache writeback and invalidations can be initiated by hardware or software.
- On the Pentium processor, each of the caches are 8 Kbytes in size and each is organized as a 2-way set associative cache. There are 128 sets in each cache, each set containing 2 lines (each line has its own tag address). Each cache line is 32 bytes wide.



- In the Pentium processor, replacement in both the data and instruction caches is handled by the LRU mechanism which requires one bit per set in each of the caches.
- The instruction and data caches can be accessed simultaneously. The instruction cache can provide up to 32 bytes of raw opcodes and the data cache can provide data for two data references all in the same clock.
- This capability is implemented partially through the tag structure. The tags in the data cache are triple ported. One of the ports is dedicated to snooping while the other two are used to lookup two independent addresses corresponding to data references from each of the pipelines.
- The instruction cache tags of the Pentium processor are also triple ported. Again, one port is dedicated to support snooping and the other two ports facilitate split line accesses (simultaneously accessing upper half of one line and lower half of the next line).
- The storage array in the data cache is single ported but interleaved on 4-byte boundaries to be able to provide data for two simultaneous accesses to the same cache line.
- Each of the caches are parity protected. In the instruction cache, there are parity bits on a quarter line basis and there is one parity bit for each tag.
- The data cache contains one parity bit for each tag and a parity bit per byte of data.

	Instruction cache								Data cache							
Directory Entry structure	P	Tag/Page(A31:A12)					S/I		P		Tag/Page(A31:A12)				MESI states	
Cache Line structure	P	QW	P	QW	P	QW	P	QW	DW	DW	DW	DW	DW	DW	DW	DW
	------								Each DW has:							
									P	Byte	P	Byte	P	Byte	P	Byte
32 bit address interpretation	tag/page (20 bits)			Index (7 bits)			Byte (5bits)		Tag/page (20 bits)		Line (7 bits)		Bank select (3 bits)		XX (2 bits)	

- Each of the caches are accessed with physical addresses and each cache has its own TLB (translation lookaside buffer) to translate linear addresses to physical addresses. The TLBs associated with the instruction cache are single ported whereas the data cache TLBs are fully dual ported to be able to translate two independent linear addresses for two data references simultaneously.
- The tag and data arrays of the TLBs are parity protected with a parity bit associated with each of the tag and data entries in the TLBs.
- The data cache of the Pentium processor has a 4-way set associative, 64-entry TLB for 4- Kbyte pages and a separate 4-way set associative, 8-entry TLB to support 4-Mbyte pages.
- The code cache has one 4-way set associative, 32-entry TLB for 4-Kbyte pages and 4-Mbyte pages which are cached in 4-Kbyte increments.
- Replacement in the TLBs is handled by a pseudo LRU mechanism (similar to the Intel486 CPU) that requires 3 bits per set.

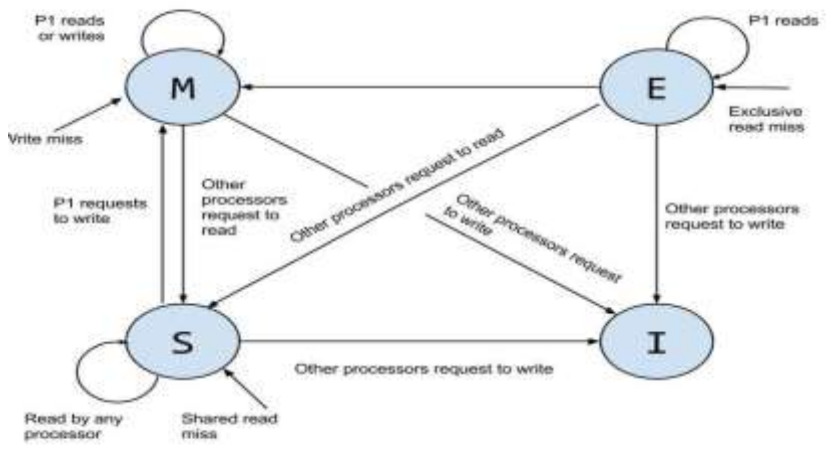
11. MESI Model (Cache Consistency Protocol):

UQ: Explain the cache consistency protocol of Pentium Processor

Cache consistency on the Pentium processor is maintained using the MESI protocol. The protocol is used to decide if a cache entry should be updated or invalidated
The data cache supports the Cache Consistency Protocol, which is a set of rules by which states are assigned to cached entries or lines. The protocol consists of four states that define whether a line is valid (HIT or MISS), if it is available in other caches, and if it has been modified. The four states, which make up what is referred to as the MESI protocol, are the M (Modified), E (Exclusive), S (Shared) and the I (Invalid) states.
The following is a description of each state:

- **An *M-state* line** is modified meaning that it is different from main memory. An M-state line can also be accessed (read/written to) without sending a cycle out on the bus.
- **An *E-state*** line is not modified. An E-state line can also be accessed (read/written to) without generating a bus cycle, with a write causing the line to become modified.
- **An *S-state*** indicates that the line is potentially shared with other caches meaning that the same line may exist in more than one cache. Reading from this line does not generate bus activity however a write will generate a write-through cycle on the bus and may also invalidate this line in other caches. A write to an S-state line updates the cache.
- **An *I-state*** indicates that the line is not available in cache. Reading from this line may cause a MISS and cause the processor to execute a LINE FILL where the whole line is fetched from main memory and placed back into cache. Writing to an INVALID line causes the processor to execute a write-through cycle on the bus.

The other piece of L1 cache, the code side, supports a subset of the MESI protocol, the S (Shared) and I (Invalid) states in order to prevent code from accidentally being corrupted since it is inherently write protected.


P1 reads or writes
M
P1 reads
E
Write miss
Exclusive read miss
P1 requests to write
Other processors request to read
Other processors request to read
Other processors request to write
Other processors request to write
S
I
Other processors request to write
Read by any processor
Shared read miss