Knowledge Update Course
for Secondary School Computer and IT Teachers

Computer Organization

Day One

Jasper Wong
email: icjwong@polyu.edu.hk

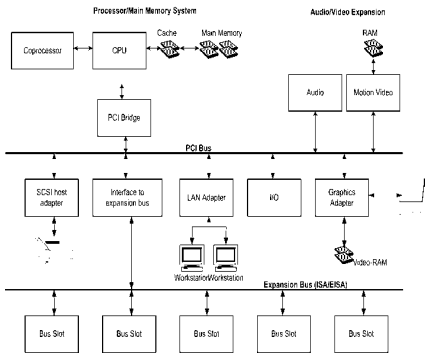
Industrial Centre
The Hong Kong Polytechnic University

July 22, 2003

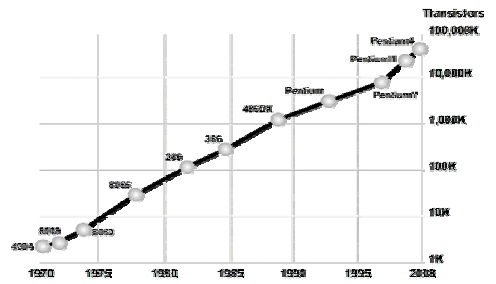
Day One Agenda

- Computer Architecture and Components
- Computer Instruction Cycles
- CISC/RISC processors
- Pipelining
- Memory
- Instructions
- Addressing Modes

Computer System Overview



Development of Intel processors



Computer Processors

Development of Intel Computer Processors

Type/ Generation	Year	Data/Addr (bit)	Level1 Cache (KB)	Memory Speed (MHz)	Internal Clock Speed (MHz)
8088 /1	1979	8/20	None	4.77-8	4.77-8
8086 /1	1978	16/20	None	4.77-8	4.77-8
80286 /2	1982	16/24	None	6-20	6-20
80386DX /3	1985	32/32	None	6-33	6-33
80386SX /3	1988	16/32	8	6-33	6-33
80486DX / 4	1989	32/32	8	25-50	25-50
80486SX /4	1989	32/32	8	25-50	25-50
80486DX2 /4	1992	32/32	8	25-40	25-80

Computer Processors

Development of Intel Computer Processors

Type/ Generation	Year	Data/Addr (bit)	Level1 Cache (KB)	Memory Speed (MHz)	Internal Clock Speed (MHz)
80486DX4/4	1994	32/32	8+8	25-40	75-120
Pentium /5	1993	64/32	8+8	60-66	60-200
MMX /5	1997	64/32	16+16	66	166-23
Pentium Pro/6	1995	64/36	8+8	66	150-200
Pentium II/6	1997	64/36	16+16	66	233-300
Pentium II/6	1998	64/36	16+16	100	450-1.2GHz
Pentium III/6	1999	64/36	16+16	266	500-1.67GHz
Pentium 4/7	2000	64/36	12+8	400	1.4GHz-2.2GHz

Intel 386/486 processors

- 80386SX is half speed of DX
- Intel 486, twice faster, 8K cache on chip
- 486DX with co-processor on board
- 486DX2, double internal clock for on chip data transfer
- 486DX4, triple speed, 16k cache

Pentium Processor

- Year 1993, CISC, 3.3 million transistors, 0.35-micron process
- Internal 32bit bus, external 64bit data bus
- Majority 3.3v, dual pipelined superscalar, execute more instructions per clock cycle
- Prefetch, Instruction Decode, Address Generate, Execute and write Back
- Two parallel integer pipelines, enabling it to read, interpret, execute and dispatch two operations simultaneously
- Two 8KB, two-way set, associative buffers (Level 1 cache)
- One for instruction and one for data

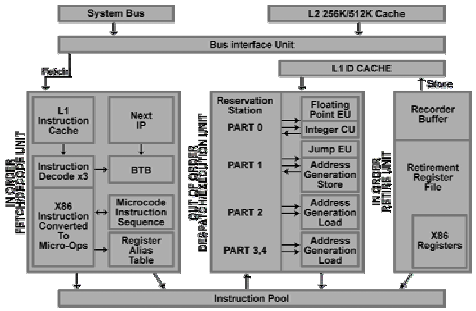
Pentium Processor

- A Branch Target Buffer (BTB) provides dynamic branch prediction
- BTB enhances instruction execution by “remembering” the way an instruction branch and applying the same branch the next time the instruction is used
- 80-ponit Floating Point Unit Handles “real” numbers
- System Management Mode (SMM) controls power use of the processor and peripherals

Pentium Pro Processor

- Year 1995, CPU 5.5 million transistors , Level2 cache 15.5 million transistors, aimed high end server and workstations
- High order superscalar optimizes for 32bit operations
- On-chip L2 Cache between 256KB and 1MB at clock speed
- On chip L2 Cache enables 64-bit rather than 32bit data path
- “Dynamic Execution” improves performance, includes branch prediction, data flow analysis and speculative execution
- Utilize wasted clock, making predictions about program flow to execute instructions in advance

Pentium Pro Processor
(for reference)



Pentium Pro Processor

- First x86 family employs super pipelining, 14 stages pipeline
- 8 stages fetch/decode unit for decoding and issuing instructions
- 3 stages out-of-order dispatch/execution unit for executing instructions
- 3 stages in-order retirement unit
- Convert CISC x 86 instructions to RISC micro-ops

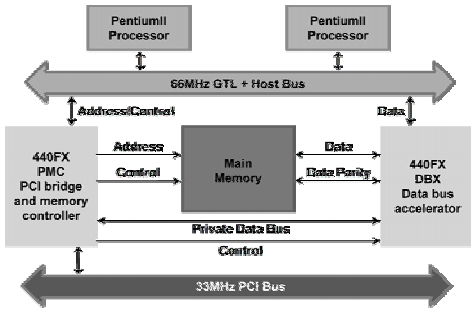
Pentium MMX Processor

- P55C MMX Launched at 1997 with Multimedia eXtension
  - Double on-board Level1 cache to 32KB
  - 57 new instructions for video, audio and graphical data
  - A new process called Single Instruction Multiple Data (SIMD) enabled one instruction to perform the same function on multiple pieces of data simultaneously
- 32KB cache, speed up data retrieval from L2 cache
- 8 SIMD and eight enhanced(64bit) registers enhanced parallel processing, 8 byte data processed in on clock cycle
- Benefits multimedia and graphics applications

Pentium II Processor

- Launched at mid-1997, major changes:
  - Dedicated bus for Level 2 cache, slot1 package
  - Processor, Level 2 cache and heat sink mounted on a small board
  - Pentium Pro/MMX hybrid – Dual Independent bus (DIB) + MMX
  - 2.8V operation, decrease power at higher frequencies
  - Speculative Execution increases the speed of instructions, looking ahead of the current instructions and processing further instructions likely executed
  - Support 2 processors by a gunning-transceiver-logic (GTL+)
  - Chipset imposes multiprocessors limitation
  - PII tackled 16 performance, Pentium Pro good for 32-bit Windows NT but bad for 16-bit Windows 95, PII solved this by segment descriptor cache
  - Fast floating point arithmetic and Accelerated Graphics Port (AGP)

Pentium II Processor



Pentium II Processor

- Dual Independent Bus (DIB)
  - Enable PII access data buses simultaneously and in parallel
  - Processor access data from L2 cache using high speed backside bus, it separates from CPU to main memory system bus (front side bus)
  - Bus between CPU and L2 cache runs half processor clock, it is scalable
  - Front side bus can increase speed without affecting L2 cache bus
  - Pipelined front side bus also enables multiple simultaneous transactions
  - DIB provide 3 times of a single bus bandwidth

Pentium II Processor

- Deschutes, two CPU lines:
  - Same as slot 1 PII, but using 0.25-micron technology for notebook
  - 0.35-micron with speed 233MHz to 400MHz, 66MHz external bus speed
- 1998 launched a new Deschutes
  - 100MHz system bus width, 450MHz processor speed
- Pentium II Xeon processor
  - Slot2 for very high end servers for high performance
  - Slot1 designed for volume production

Celeron Processor

- Launched at 1998 for low cost market
- Based on PII, 0.25-micron, include AGP, ATA-33, SDRAM and ACPI technologies
- Initially, 266 & 300MHz with no L2, market no good
- Later, 300AMHz with 128KB on-die L2 at full CPU speed
- Two form factors: Sing Edge Cartridge (SEPP) compatible to slot 1 & Plastic Pin Grid Array (PPGA) pin 370 socket
- New Celeron launched at 2000, 0.18-mciron, low cost Flip-chip pin grid array (FC-PGA) packaging and required new FC-PGA socket-370
- Introduction of Coppermine-based Celeron ran at 566MHz, and at early 2002, higher speed at 1.8GHz

Pentium Xeon Processors

- Launched at 1998 at 400MHz, represents a combination of Pentium Pro and PII for performance workstation and server markets
- Slot2, increase L2 cache, 512KB or 1MB, support ECC SRAM
- L2 runs full sped and same as CPU core speed
- Can run multiple processors
- PIII Xeon launched at 1999 with new Streaming SIMD Extensions (SSE) instruction set added
- Pentium 4 Xeon launched at 2001 with clock speeds 1.7GHz
- microPGA Socket 603, 2x64 PCI buses, Memory Repeater Hubs (MRH-R), Max memory 4GB, integrated with L3 cache of 512L3 cache of 512KB and 1MB

Pentium III Processors

- Launched in 1999, 50 new SIMD Extensions for improving floating-point performance
- 8 new 128-bit floating-point registers, can lead to 4 floating-point results returned at each cycle,
- 12 New Media instruction for further support of multimedia data processing,
- 8 instructions for New Cacheability instructions, and improve CPU's L1 cache efficiency
- Unless for 3D/games applications, performance is similar to PII
- Pentium III using "coppermine" launched in Oct 1999, using 0.18-micron process

Pentium III Processors

- Smaller die size, lower operating voltage, power-efficient system design
- New PIII L2 size is 256KB, but full speed, enhanced cache "Advanced Transfer Cache" (ATC), 256-bit wide bus, Advanced Buffering technology which increases buffers between processor and system bus
- In July 2000, Intel recalled all 1.13GHz CPU due system malfunction for some applications
- New processor core with 0.13-micron with enhancing Data Prefetch Logic (DPL)
- DPL analyses data access patterns and uses available bandwidth to "prefetch" data into L2 cache
- Assisted Gunning Transceiver Logic+ (AGTL) signaling at 1.25V, 1 G bandwidth

Pentium 4 Processors

- Hyper Pipeline comprises 20 pipeline stages
- L1 cache comprising an Execution Trace Cache, store up to 12K of decoded x86 instructions
- Rapid Execution Engine that pushed processor's ALU to twice the core frequency resulting in higher throughput
- 3 separate clocks: core frequency, ALU frequency and bus frequency
- Out-of-order speculative engine
- 256KB L2 Advanced Transfer Cache
- SIMD Extension 2 (SSE2)
- First 400MHz system bus
- Support for Direct Rambus DRAM (DRDRAM)
- Spring 2002, 3GHz
- I850 chipset for DRDRAM, or DDR SDRAM supporting chipsets i845 from SiS & VIA

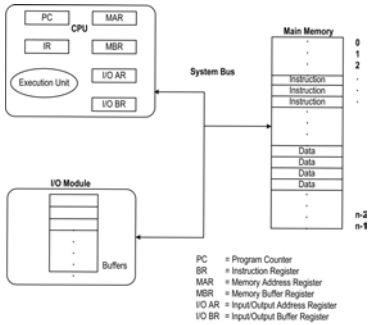
Itanium Processors

- Intel's newest microprocessor , to be released
- 64-bit architecture (IA-64)
  - 128x64-bit general purpose registers
  - 128 x 82-bit floating point registers
- Explicit parallelism
  - Packaged in 128-bit bundles ready for execution, make up of three 41 bit instructions plus 5-bit template, the template will identify type of execution unit (memory, floating point , branch, general), can dispatch all 3 instructions in parallel
- Speculation
- Prediction
- 64 bit address lines

Computer System Components

- There are four major components:
  - Central Processing Unit (CPU)
  - Memory (Primary and Secondary)
    - Primary – Temporary (e.g. Random Access Memory)
    - Secondary – Permanent (e.g. Hard Disk)
  - Input and Output Device (I/O) – Peripheral e.g. keyboard and mouse
  - Buses e.g. AGP and PCI

Computer System Components



Computer system Components

- What is instruction ? (e.g. 1011 1010 0101)
- What is data?
  Both of them can be considered as binary
  Code (stored in memory).

Machine code

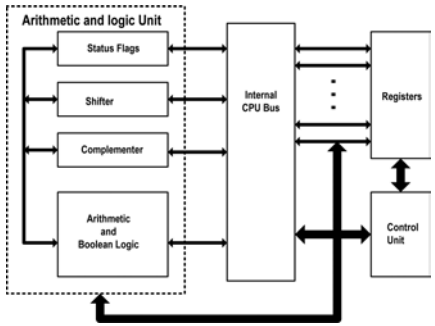
Add ax, bx

instruction

data

- Machine Cycle
  - Fetch instruction
  - Decode / Interpret instruction
  - Fetch data
  - Execute / Process data
  - Write data

Central Processing Unit



Central Processing Unit

- Major components:
  - Arithmetic and Logic Unit (ALU), which does the actual computation or processing of data.
  - Control Unit (CU), which control the operation of the processor.
    - Fetches instructions
    - Carries out instructions
    - Retrieves/Stores data from/to registers
    - Requests/Sends data from/to memory
    - Passes data and commands to the ALU, accepts results
  - Internal CPU Bus, which is responsible for data /control signal transfer between the various units (e.g. ALU, registers and control unit).
  - Registers, which are temporarily storage within the CPU (e.g. instructions and data)

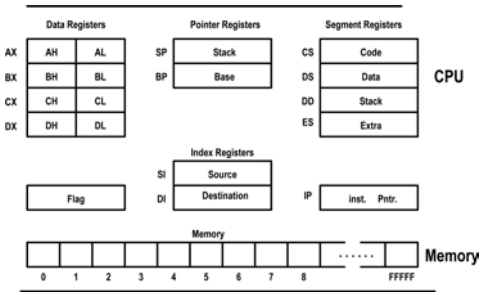
Registers

- User-visible register
  - General purpose register
  - Data register
  - Address register

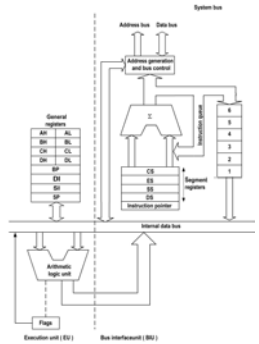
Registers

- Control and Status Register, which is used to control the operation of the CPU
  - Program Counter (PC)
    - Contains the address of the next instruction to be fetched to memory
  - Instruction Register (IR)
    - Contains the opcode (or instruction) being executed
  - Memory Address Register (MAR)
    - Specifies the address in memory of the word to be written from or read into the MBR
  - Memory Buffer Register (MBR)
    - Contains a word to be stored in memory, or is used to receive a word from memory
  - Status Register / Flag (Program Status Word, PSW)
    - Contain condition codes plus other status information.
    - e.g. Sign, Zero, Carry, Equal, Overflow, Interrupt...

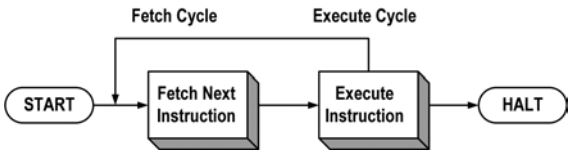
8086 Registers



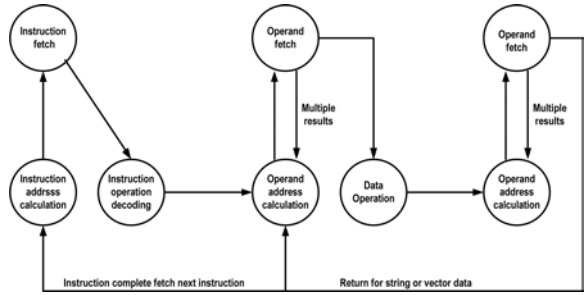
8086 Processor Model



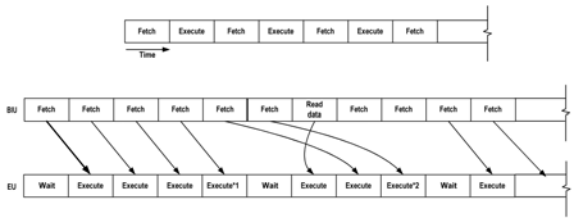
Instruction Cycle



Instruction Cycle



Non-pipeline/pipeline



1. Request a data in in a queue
2. Jump instruction occurs

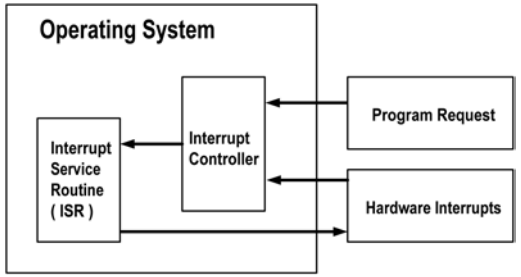
Interrupt

- Problem: If there are some urgent demand (such as data from modem), it is impossible to wait till the end of each program execution.
- Interrupt is used to stop the executing job and run other program
- Then, after execution of that specific program, it returns back to the original program.

Interrupt

- An interrupt allows a program or an external device to interrupt the execution of a program.
- The generation of an interrupt can occur by hardware (hardware interrupt) or software (software interrupt).
- When an interrupt occurs, an interrupt service routine (ISR) is called

Interrupt



PC Interrupt

Interrupt	Name	Generated by
08H	System Timer	IRQ0
09H	Keyboard	IRQ1
0AH	Reserved	IRQ2
0BH	Serial Communication (Com2:)	IRQ3
0CH	Serial Communication (Com1:)	IRQ4
0DH	Parallel Port (LPT2:)	IRQ5
0EH	Floppy controller	IRQ6
0FH	Parallel Port (LPT1:)	IRQ7
70H	Real Time Clock	IRQ8
71H	Redirection of IRQ2	IRQ9
72H	Reserved	IRQ10
73H	Reserved	IRQ11
74H	Reserved	IRQ12
75H	Math Co-processor	IRQ13
76H	Hard Disk Controller	IRQ14
77H	Reserved	IRQ15

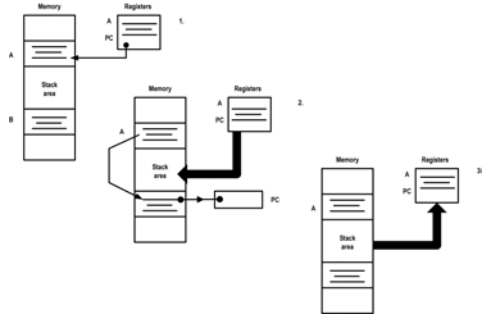
Interrupt

- Interrupt Handling
  - Generates necessary signals to fetch the next instruction from memory (the required instruction).
  - Changes the contents of PC to the next instruction
  - Determines the type of instruction in IR
  - Determines the operand's address
  - Fetches the operand & places it in the register (MAR)
  - Carries out the specified operation
  - Stores the result in proper place including memory (MBR) & register set (PC)
  - Check Interrupt and response if necessary
  - Repeat the first step

Interrupt

- Each device is connected to the interrupt controller through an interrupt request line (IRQ)
- If there is any request from these devices, the controller will inform the CPU (by using the interrupt number)
- The CPU will go to memory to check the starting address of the service routine - interrupt vector
- Save the context (e.g. registers value)
- Load and run the interrupt service routine (ISR) and go back to the original program (continue to execute the original program).

Interrupt Service Routine

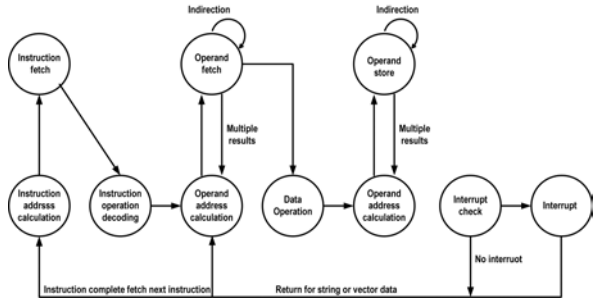


Instruction Cycle with Interrupt

- Fetch the instruction
- Decode it
- Fetch operands
- Perform the operation
- Store results
- Recognize pending interrupts

- Disadvantage:
  - Does not provide a high level of efficiency based on the sequential fashion

Instruction cycle with Interrupt



Instruction Pipeline

- Pipelining is one of the effective techniques to improve the performance of the CPU
- Perform all tasks concurrently, but on different (sequential) instructions, so the result is temporal parallelism
- An ideal pipeline divides a task into *k* independent sequential subtasks
  - Each subtask requires 1 time slot to complete
  - The task itself then requires *k* time slots to complete

Instruction Pipeline

- Fetch instruction (FI):
  - read the next expected instruction into a buffer
- Decode instruction (DI):
  - determine the opcode & the operand specifies
- Calculate operands (CO):
  - calculate the effective address of each source operand. This may involve displacement, register indirect, indirect, or other forms of address calculation
- Fetch operands (FO):
  - fetch each operand from memory. Operands in registers need not be fetched
- Execute instruction (EI):
  - perform the indicated operation & store the result in the specified destination operand location.
- Write operand (WO):
  - store the result in memory

Some textbooks Simplify the 6 stages into 4:

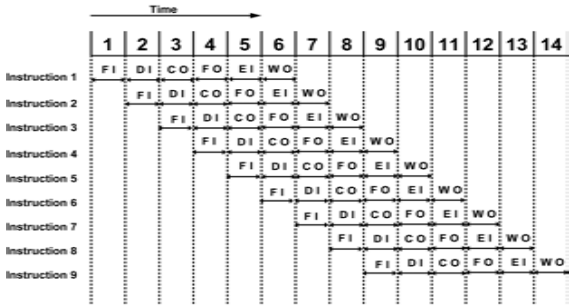
F (Fetch instruction)

D (Instruction decode)

E (Execute instruction)

W (Write operand)

Pipeline Timing



- **Pipelined execution of 9 instruction in 14 time units vs 54**

Instruction Pipeline Performance Improvement

- For n iterations of the task, the execution times will be:
  - With no pipelining: *nk* time units
  - With pipelining: *k + (n-1)* time units
- Speedup of a k-stage pipeline is thus:
  - *S = nk / [k+(n-1)]* ==> *k* (for large n)

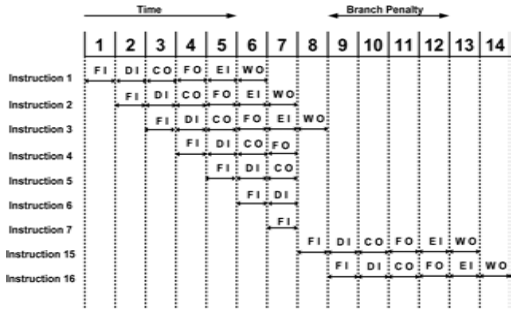
Limitations of Pipelining

- **Data Dependence**
  - Pipelining, as form of parallelism, must ensure that computed results are the same as if computation was performed in strict sequential order
  - Data dependencies limit when an instruction can be input to the pipeline
    - Data dependence examples:
      - A = B + C
      - D = E + A
      - C = G x H
      - A = D / H

Limitation of Pipelining

- **Branching**
  - For the pipeline to have the desired operational speedup, it is ideal to use long strings of instructions (no branching)
  - *Drawbacks are:*
    - 15-20% of instructions in an assembly-level stream are conditional branches
    - Of which 60-70% take the branch to a target address results in a great branch penalties

Pipeline Timing



- **Impact of a branch after instruction 3 to instruction 15**

Minimising Branch Penalty

- Multiple streams
  - Replicate the initial portions of the pipeline & fetch both possible next instructions
- Prefetch branch target
  - When the branch instruction is decoded, begin to fetch the branch target instruction
- Branch prediction
  - Make a good guess as to which instruction will be executed next & start that one down the pipeline
- Delayed branch
  - Find the valid instructions to execute in the pipeline while the branch address is being resolved (program code optimization - possibly by compiler)

CISC Processor

- Two CPU architectures dominated the PC industry
  - CISC for Intel's Pentium
  - RISC for Motorola Power PC
- Complex Instruction Set Computer (CISC)
  - CPU uses microcode to execute every comprehensive instruction set
  - Instruction set may be variable in length and use all addressing modes
  - Require multiple clock cycles to execute
  - Require complex circuitry for decoding
  - Performance bottleneck

RISC Processor

- Reduced Instruction Set Computer (RISC)
  - small or constant instruction sets
  - using fewer instructions
  - simple addressing modes
  - execute an instruction in a single machine cycle
  - result in simple silicon circuitry, better cost and performance
- Design goal for RISC chip:
  - Reduce accesses to main memory
  - Keep instructions and addressing modes simple
  - Make good use of registers
  - Pipeline everything
  - Extensively utilize the compiler

RISC Processor

- Reduce Access to main memory
  - processors are much faster than memories
  - end up with waiting for each memory access, waiting execution cycles
  - Reduce memory access by adding instruction and data cache
- Cache
  - High-speed RAM for storing instruction and data
  - Processor examine the cache first before trying to access the memory
  - Cache will supply data if the address stored in cache matches the address being addressed for memory read, called a "hit"
  - Cache (10ns) usually 10 times faster than main memory(60ns)
  - Only a "miss", cannot find the data, a full access to main memory
  - Happen once since a copy of data written into a cache after a "miss"
  - Instruction and data cache stored frequently used instruction & data

RISC Processor

- Keep instructions and addressing modes simple
  - Programmers only use a small subset of the available instructions
  - Same as instruction modes
  - Fewer instructions and addressing modes on silicon reduces the complexity of instruction decoder, the addressing logic, and the execution unit
  - This allows the machine running at faster speed and less work for each clock period
  - Pentium cannot meet this goal in order to keep backward software compatibility of the previous processors 80x86 and 80486
  - Pentium looks like a RISC machine

RISC Processor

- Make good use of registers
  - RISC machines typically have large sets of registers
  - No. of registers in a processor can affect processing performance
  - Complex calculation require the use of several data values, and hence the no of memory or registers
  - Calculation will be faster if data values stored in internal registers
  - Pentium has a fairly large set of registers including eight 80 bits floating-point registers

RISC Processor

- Pipeline everything
  - A technique used to enable one instruction to complete with each clock cycle
  - On a non-pipeline machine, it requires, say 9 clock cycles for individual fetch, decode and execute cycles

RISC Processor

- On a pipeline machine, where fetch, decode and execute operations are performed in parallel, say 5 cycles need to execute the same instructions
- The Pentium employs two types of instruction pipelines, U and V
- Pentium is very like a RSIC machine

RISC Processor

- Extensively utilize the compiler
  - High-level program converted into assembly code by a compiler
  - A Pentium can perform many optimizations on assembly code to take advantages of the Pentium's architectural advances
  - A compiler can re-order instructions for parallel instruction execution in the floating-point unit or dual-integer pipelines

RISC Processor

- Instructions can be rearranged to take advantages of branch prediction strategy, use of instruction and data cache
- Allocate the minimum no of processor registers during parsing of an arithmetic statement to reduce the no of clock cycles or machine code

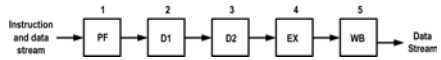
Pentium Superscalar

- Processor capable of parallel instruction execution of multiple instructions are known as superscalar machines
- Pentium is capable of executing two integer or two floating-point instruction through its twin U and V pipelines
- 4 restrictions on parallel execution of 2 integer instructions
  - Both instructions must be simple
    - May take one clock cycle, register-to-register instruction, MOV, INC, DEC..
    - May take 2 or 3 clock cycles, ALU instruction that use both register & memory
    - Near conditional jumps: JZ, JNZ

Pentium Superscalar

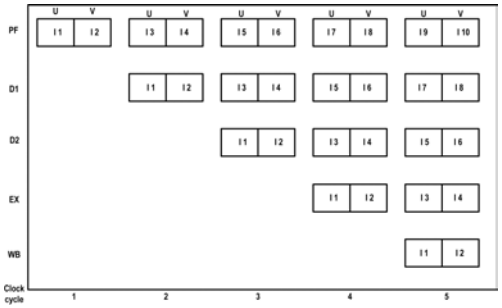
- No data dependencies existing between the instructions
  - Read-after-write dependency
  - Both instruction write to same operand e.g. MOV AX, BX ; MOV AX, CX
- No immediate data and a displacement value in the instructions
  - e.g. MOV, TABLE[SI], 7
- Prefixed instructions, e.g. MOV ES:[DI], AL, may only execute in U pipeline

Pentium Pipelining



- Five-stage U and V instruction pipelines, each pipeline has the following stages:
  - PF Prefetch
  - D1 Instruction Decode
  - D2 Address Decode
  - EX Execute, Cache and ALU access
  - WB Writeback

Pentium Pipelining



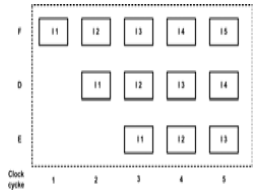
Pentium Pipelining

- U pipeline can execute any processor instruction
- V pipeline only executes simple instructions
- prefetched from cache or memory
- Decode in each D1 of U & V, check execution pairing
- Additional clock in D1 for a prefix byte instruction and may not execute in U pipeline
- Operands addresses calculated in stage D2
- In EX stage, operands read from cache/memory, perform ALU operations, and branch predictions
- In WB stage, write completed results, verify branch predictions

Dynamic Execution

- A new approach to processing software instructions that reduce idle processor time to an absolute minimum
  - Multiple Branch Prediction -- look ahead of instruction for branching instruction
  - Data Flow Analysis -- look at upcoming software instructions and determines if they are available for processing dependent on other instructions, optimize sequence for processing and begin executing these instructions
  - Speculative Execution – allow processor to execute instructions in a different order from which they entered the processor. This “out-of-order” execution allows the processor to execute instructions in an efficient manner. The results of these instructions are stored as speculative results until their final states can be determined.

Pentium Branch Prediction

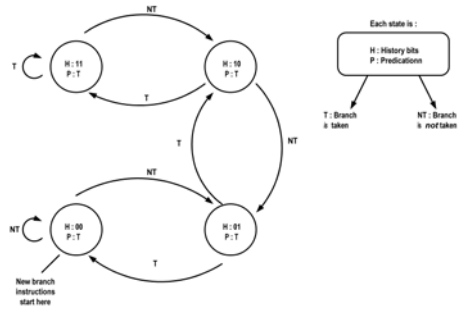


- Transfer instruction in pipeline affects performance
- Say, I_3 conditional jump to I_{50}
- I_4 & I_5 Instruction in pipeline becomes invalid
- I_4 & I_5 occur after I_3, discarded or flushed
- Pipeline loads I_5
- New instructions begin with I_5
- Dynamic branch prediction scheme for solving this problem

Pentium Branch Prediction

- **Dynamic branch prediction scheme**
  - Prediction on branching instruction currently in pipeline
  - If prediction is true, pipeline will not be flushed, no clock cycle lost
  - If prediction is false, pipeline is flushed, correct instruction starts over
- **Branch target buffer (BTB)**
  - Special cache stores instruction & target addresses of any branch instructions
  - Along with instruction address, BTB stores 2 history bits for last 2 execution instructions
  - Repeated history bits failure results in prediction false
  - Two 32bits prefetch buffers work with BTB & D1 stage of U & V
  - One buffer for current program address, one for prefetch instruction from target address when prediction is true
  - BTB accesses the branch address during D1, if prediction is true, 2nd buffer is active and no cycles lost
  - If prediction is false, 3 clocks in U & 4 clocks in V are lost
  - Usually Conditional jumps applied in loop, and prediction is mostly true, min cycles lost

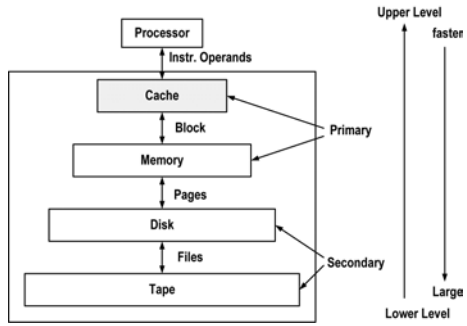
Pentium Branch Prediction



Memory

- Memory is divided into 8 bit units called bytes
- The byte is the smallest addressable unit of memory
- Each byte has a fixed numeric address – a binary integer assigned to each byte consecutively starting with zero
- Assembler and high-level language variable names are translated to numeric addresses
- 1 Kilobyte = 2^{10} bytes = 1024 bytes
- 1 Megabyte = 2^{20} bytes
- 1 Gigabyte = 2^{30} bytes

Memory



Memory

Types	Speed
Conventional/Fast Page	4.77 MHz (XT) 12 MHz (AT)
EDO / BEDO	33 MHz
SDRAM	Up to 133 MHz
DDR RAM	266 MHz up
RDRAM	Up to 400 MHz

Note: Must support by CPU and Chipset

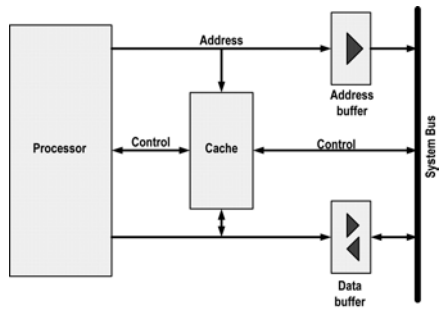
Memory

CPU registers	15 - 30 nsec
Cache Memory	50 - 100 nsec
Conventional Memory	75 - 500 nsec
Hard disk	10 - 50 msec
Floppy disk	95 msec
CD-ROM	100 - 600 msec

Cache Memory

- Cache memories are one of the most effective techniques that computer architects have for reducing average latency
- By storing frequently accessed data in small, fast memories located physically close to the processor, the latency of most memory references can be greatly reduced
- A “hit” is said to occur when the required data is found in the cache, a “miss” when it is not.
- The capacity of a cache is simply the amount of data that can be stored in the cache
- The line length describes the size of the units of data that the cache operate on.
- Long cache lines tend to increase the hit rate of the cache by fetching more data into the cache on each cache miss, but they can increase the total execution time of program by increasing the amount of unused data that gets fetched.

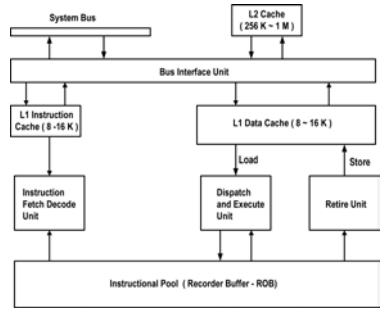
Cache Memory



Cache Memory

- Cache size between 1K and 512K bytes would be effective
- The performance of the cache is very sensitive to the nature of the work load, it is impossible to arrive a single “optimum” cache size.

Pentium III Cache Example



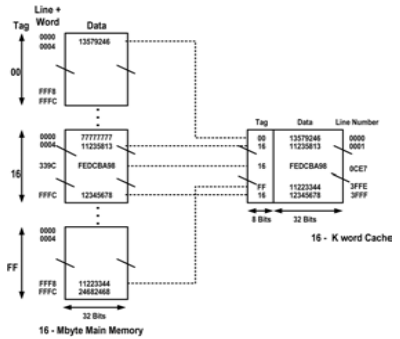
Cache Memory

- Mapping to the main memory
  - Direct Mapping
  - Associate Mapping
  - Set Associate Mapping
- Replacement Policy
  - Least recently used
  - First in First out
  - Least frequently used
  - Random

Direct Mapping

- Maps each block of main memory into only one possible cache line
- Seeks to avoid look-up by deriving the cache entry corresponding to the referenced memory address
- Simple & inexpensive to implement
- But results in a fixed cache location for any given block

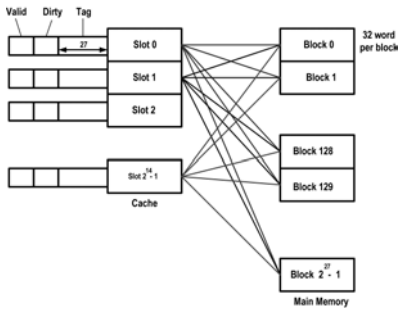
Direct Mapping



Associate Mapping

- Permits each main memory block to be loaded into any line of the cache.
- The cache control logic interprets a memory address simply as a tag & a word field
- The tag field uniquely identifies a block of main memory
- Complex circuitry required to examine the tags of all cache lines in parallel

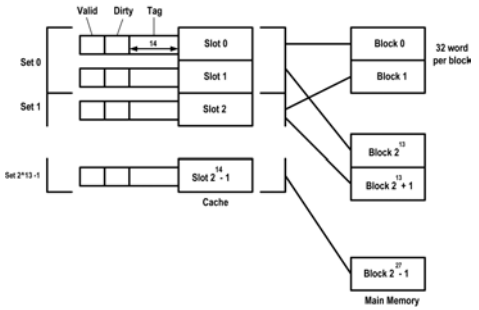
Associate Mapping



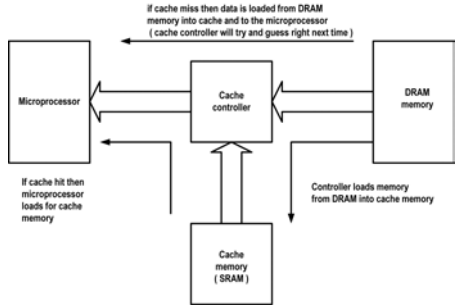
Set Associate Mapping

- Exhibits the strengths of both the direct & associative approaches
- The cache control logic interprets a memory address simply as 3 fields: tag, set & word

Set Associate Mapping



Cache Operation



Replacement Policy

- When a line must be evicted from a cache to make room for incoming data, either because the cache is full or because of conflicts for a set, the policy is to determine which line is evicted.
- The perfect replacement policy would examine the future behavior of the program being run and evict the line that results in the fewest cache misses.
  - Least-recently used (LRU), higher hit rate, but relatively complex to implement, applied in two-way associative caches
  - Not-most-recently used, low hardware cost, applied in more-associative caches
  - Random replacement, in which randomly selected line from the appropriate set is evicted to make room for incoming data
  - First in, First out

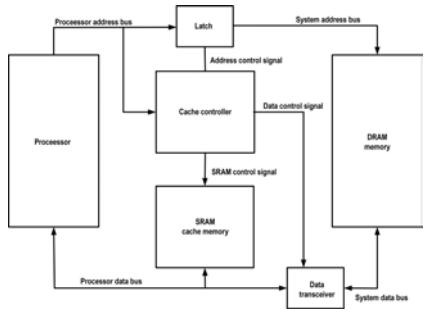
Replacement Policy

- Write-back cache store modified data in the cache, writing it out to the next level of the memory hierarchy only when the line is evicted
  - Write-back caches give higher performance, because most of lines that are written multiple times before they are evicted from the cache

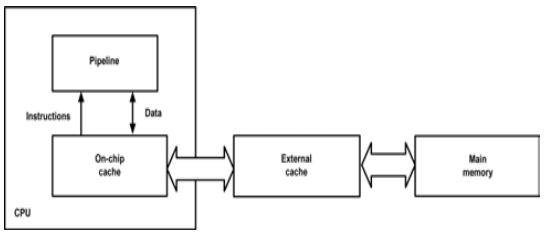
Replacement Policy

- Write-through caches send each write to the next level of the hierarchy when it occurs
  - Write-through caches are somewhat simpler to design and are sometimes used when another device is allowed to access the next level of the memory hierarchy, because they keep the contents of the next level of the hierarchy consistent with the cache at all times.

Write-through Cache



Instruction and Data Caches



Instruction and Data Caches

- High speed memory for speed up memory access and reduce processor busses traffic
- On-chip cache feeds instructions and data to CPU's pipeline
- On-chip cache searched first for instruction or data
- "hit" if in cache, "miss" if not in cache
- A copy of instruction or data will written from memory to cache
- Access time will be in between "hit" and "miss" access time
- Hit ratio specifies the percentage of hit ratio, or performance
- Hit ratio depends on: program size, type and amount of data used by the program, addressing activity during execution

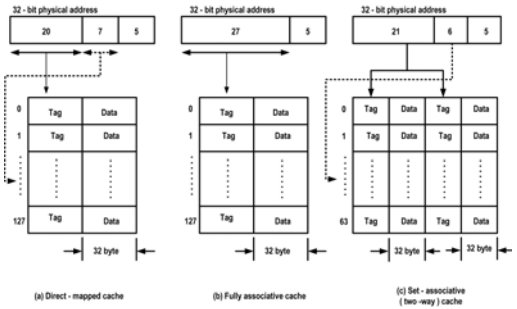
Instruction and Data Caches

- Program characteristics for improving cache performance
  - Good chance to access the same location again
  - Good possibility to access the next location
- These accesses usually maintain a locality of reference in a small range of memory
  - MOV CX, 1000
  - SUB AX, AX
  - NEXT: ADD AX, [SI]
  - MOV [SI], AX
  - INC SI
  - LOOP NEXT

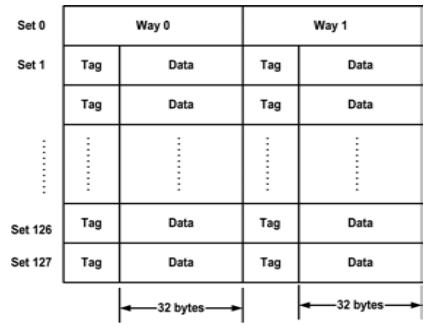
Instruction and Data Caches

- Four instructions in the loop run 1000 times
- one “miss”, 999 “hit”, speed up tremendously
- When a “miss” occurs, a group of locations is copied to cache
- The group “line of data” prepares for cache for “hits”

Cache Organization

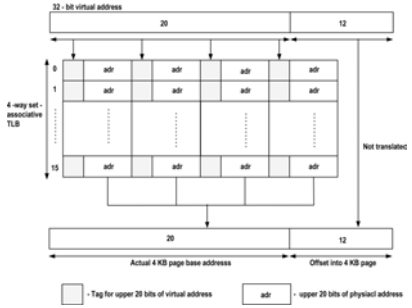


Instruction and Data Caches



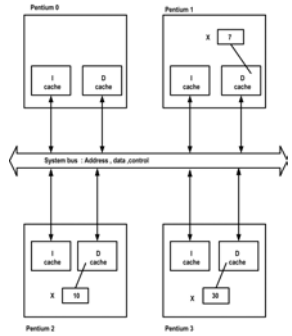
Translation Lookaside Buffers (TLBs)

(for reference)



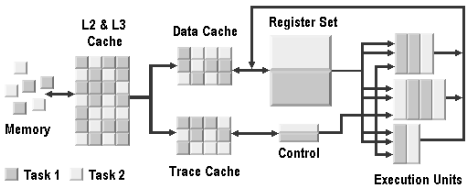
Cache Coherency in a Multiprocessor System

(for reference)



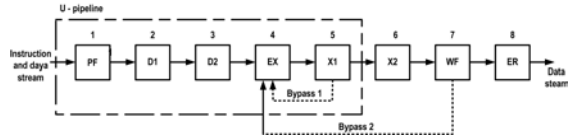
- Mechanism for maintaining Cache Coherency (MESI) with 2bits to store the states
- Modified: current line modified (does not match main memory), is available in a single cache.
- Exclusive: current line not modified (match main memory), available in a single cache, writing changes state to modified.
- Shared: exits in more than one cache, A write to this line causes a writethrough to main memory and may invalidate copies in other caches
- Invalid: current line is empty, a read generates a "miss", a write will cause a writethrough to main memory

Hyper-Threading Technology



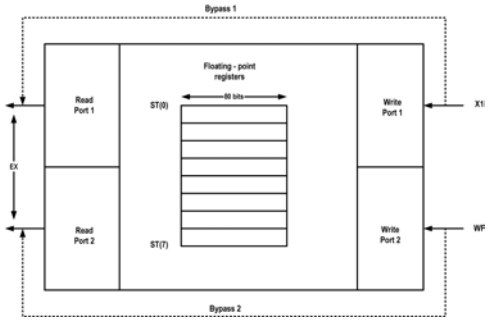
Architecturally, a processor with Hyper-Threading technology is viewed as consisting of two logical processors, each of which has its own IA-32 architectural state. After power up and initialization, each logical processor can be individually halted, interrupted, or directed to execute a specified thread, independently from the other logical processor on the chip. The logical processors share the execution resources of the processor core, which include the execution engine, the caches, the system bus interface, and the firmware.

Floating-point Unit Pipeline

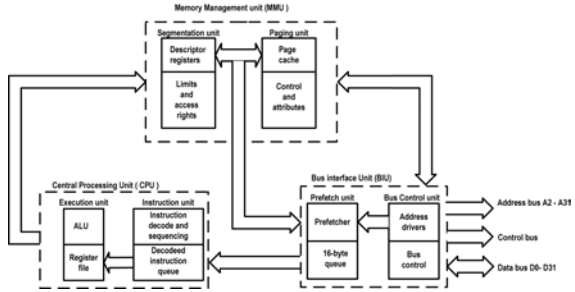


- PF Prefetch
- D1 Instruction decode
- D2 Address generation
- EX Memory & register read, FP data converted to memory format, memory write
- X1 FP execute, stage one, Memory data converted into FP, write operand to FP Register file, bypass2 (send data back to EX stage)
- WF Round FP result and write to FP register file, bypass 2 (send data back to EX)
- ER Error reporting, update status word

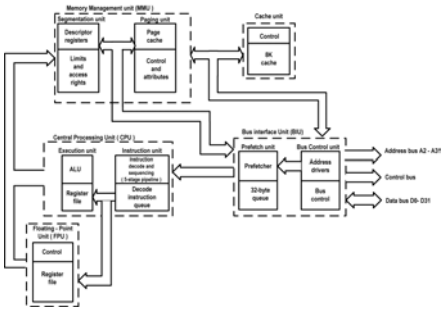
Floating-point Register File



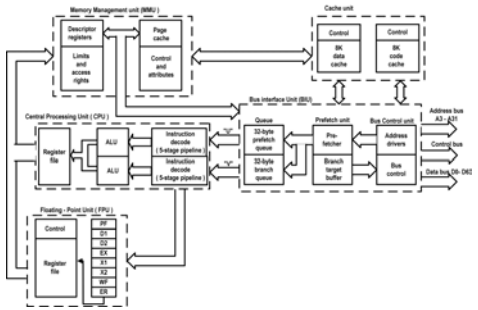
8086 Processor



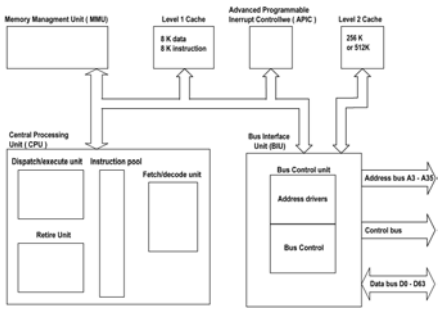
80486 Processor



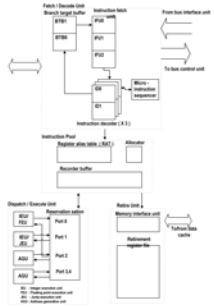
Pentium Processor



Pentium Pro processors



Pentium Pro Processor



- BTBs examine instruction stream and advise the BIU to fetch out-of-order instructions
- IFUs assemble and align information for the decoders
- 3 decoders break instructions into uniform (RISC-like) micro-ops
- Output 3 micro-ops per clock
- Logical registers (e.g. EAX) mapped to alias registers via RAT. Allocator adds status info before passing instructions to –re-order buffer
- Dispatch/Execute unit selects micro-ops from instruction pool depending on their readiness for execution. If so, Reservation station sends the micro-ops for execution
- Retire unit puts the original program back together by examining the instruction pool for completed micro-ops. These then retired in the order of the original program

Main Memory

- Mask-programmable ROM
- One-Time programmable ROM
- UV-light Erase PROM
- Electrically Erasable PROM (EEPROM)
- Flash Memory
- Static RAM
- Dynamic RAM (DRAM): DRAM, FPM DRAM, EDO DRAM, BEDO DRAM, SDRAM, PC133 SDRAM, DDR DRAM, Direct RDRAM
- Memory Modules: SIMMS, DIMMs and RIMM

Static RAM and Dynamic RAM

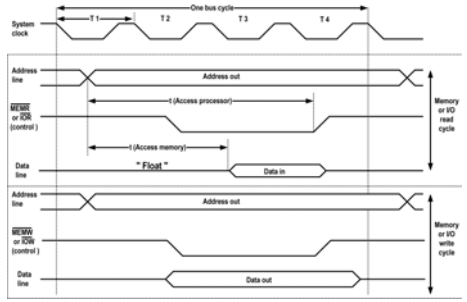
- Main difference between SRAMs and DRAMs is how their bit cells are constructed
- SRAM bit cell consists of two inverters connected in a back-to-back, and the stored data can be kept unless no power supply
- DRAM bit cell is made of capacity, instead of inverters, the stored data can only be kept with data refreshed.
- The stored data in the DRAM is not stable, as leakage current will cause the charge stored on the capacitor to drain away and be lost
- The DRAMs take up less space than SRAM

Fast Page Mode (FPM)

- One weakness of memory chip design is that an entire row's contents are sent to the multiplexer during each operation, but only 1 bit out of the row is actually sent to the output
- If the contents of the row could be kept in or near the multiplexer, it would be possible to read other bits within the same row by just sending a different column address to the DRAM, rather than doing a full RAS-CAS cycle
- Page-mode DRAMs add a latch between the outputs of the bit cells and the multiplexer. When a row address is sent to the DRAM, the entire contents of the row are stored in the latch
- This allows subsequent accesses that reference a column within the same row to simply send a second column address to the DRAM, it greatly reduces the time required to fetch a contiguous block of data from the DRAM

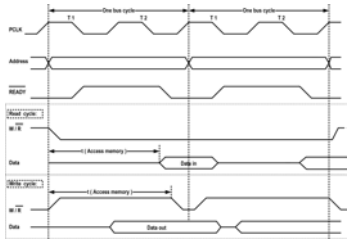
80x86 Read/Write Bus Cycles

(for reference)



386,486 & Pentium Read/Write Cycles

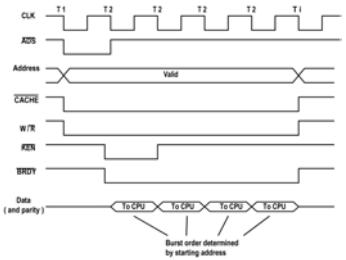
(for reference)



- t (access processor) > t (access memory) for proper function
- If not, 3 solutions: 1. decrease system clock, 2. using faster memory, 3. add in wait states

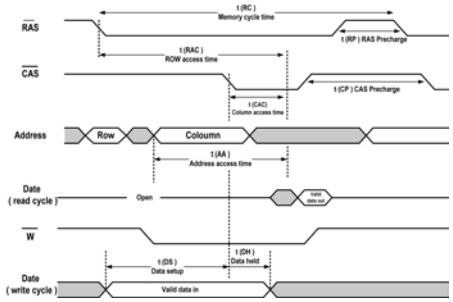
Burst Read Cycle

(for reference)

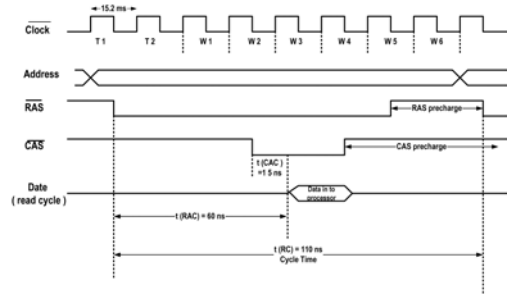


- Begin with 2 T-states, subsequence cycles transfer one quantity of data (4 bytes for 486, 8 bytes for Pentium) per clock pulse
- Processor can generate subsequent address automatically within one T-state
- Data rate doubled
- For 486, 16bytes can be transferred per burst cycle
- For Pentium, 32 bytes can be transferred per burst cycle.

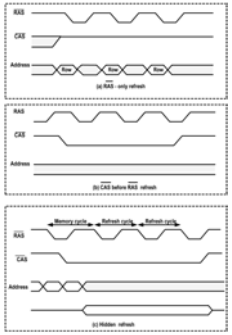
Dynamic RAM timing
(for reference)



DRAM with Wait State Timing
(for reference)

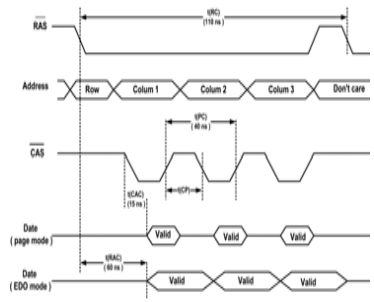


DRAM Refresh Method
(for reference)



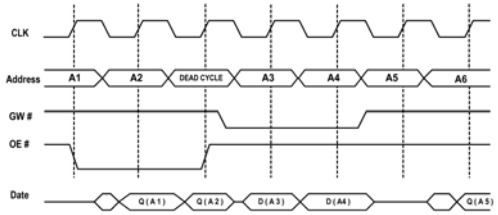
- RAS-only refresh
  - Row address sent to DRAM
  - CAS clock high
  - No data output
  - Selected row bits refreshed
- CAS before RAS refresh
  - DRAM generate refresh address internally
  - CAS active before RAS
  - DRAM generate row address and refresh all the cells
- Hidden refresh
  - After normal read or write
  - CAS clock low
  - RAS applied and incremented internally

Page-mode and EDO DRAM Timing
(for reference)



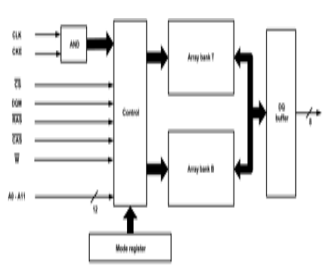
- FPM
- Most PC designed for high speed caches
- RAS clock falls, all data in a row ready for access
- CAS can consecutively selects memory cell for the page
- Access time depends on tCAC (15ns) rather than tRAC(60ns)
- EDO (Extended Data Out)
- CAS precharge time, output data is turned off
- To maintain reasonable data-out window, the width of CAS low must be extended, effect performance
- EDO keeps data valid during precharge, this can cycle time can then be reduced

SRAM Read/Write cycles
(for reference)



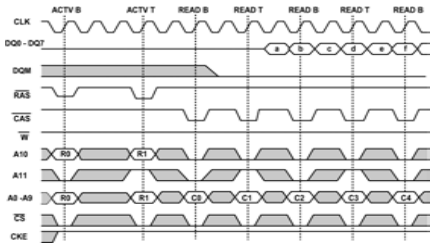
- Data are transferred on the rising edge of each clock pulse
- OE# enables the output for read cycles
- GW# performs a similar function for write cycles
- One "dead" cycle is required to switch from read to write mode

SDRAM 1M X8 Memory Bank



- SDRAM applied rising clock edge and data appeared at subsequent clock edge
- Operate in burst mode and internal burst counter
- Starting address from Processor, on board counter supplied burst group
- Interleaving mode allows memory bank begin an access while another is finishing
- One data transfer per bus
- Steps required to program mode register, bank activation, issue read command with column address
- Define burst byte no and column sequence

Synchronous DRAM(SDRAM) Timing
(for reference)



- Operated at 100MHzwith clock period 10ns
- Column interleaving
- Two bytes read from each column address , a & b, c & d

Synchronous DRAM (SDRAM)

- Typical access time 8 to 15ns, 168-pin DIMM, 64bits data width
- Work in burst mode and with a synchronous clock rate for the motherboard and system, not with CAS & RAS timing, achieved by an on-chip burst counter.
- Transfer data at burst rates up to 100MHz.
- Difficult to exceed 100MHz due to legacy SDRAM architecture
- For command read/write, SDRAM do use CAS, RAS, WE & CE signals
- Two type of SDRAM
  - EEPROM storing memory parameters for chipset to optimize the performance
  - No EEPROM, optimum values need to be set manually during BIOs setup

Synchronous Graphic RAM (SGRAM)

- Used for memories on graphics cards
- Work similar to SDRAMs
- SDRAMs are optimized for highest possible memory capacity
- SGRAMs are optimized for the fastest possible data transfer
- SGRAMs know about instructions that SDRAMs know nothing about, such as block write and write per bit

Double Data Rate DRAM (DDR-DRAM)

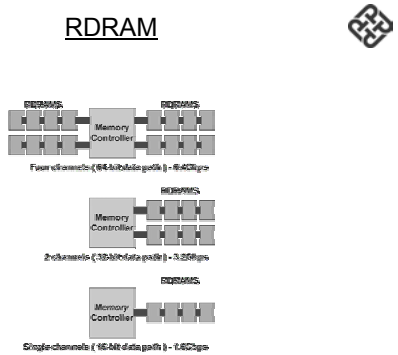
- Initial support to GeForce 256 3D graphics, no support from Intel, supported by AMD
- Double data rate by transferring data on both rising and falling clock edge, effectively double clock frequency, no actually increasing the frequency.
- DDR-DRAM is backwards-compatible
- Lead to PC-333 modules
- Naming based on peak bandwidth, not their clock rates

PC100 SRAM	PC133 SRAM	PC1600 DDR	PC2100 DDR	PC2700 DDR
8bytes x 100M =800MBps	8bytes x 133M =1.1GBps	8bytes x 200M =1.6GBps	8bytes x 266M =2.1GBps	8bytes x 333M =2.7GBps

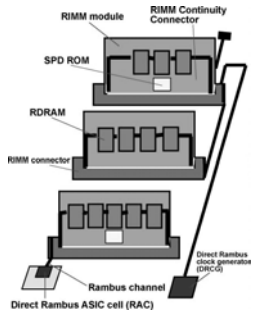
Rambus DRAM (RDRAM)

- Developed by Intel and Rambus, used first with the chipset i820
- Major elements: master device with Rambus ASIC Cell and Rambus Controller, Direct memory clock generator
- Special RDRAM bus delivers address and control information using an asynchronous block-oriented protocol
- Data rate is 1.6Gbps, operate with max clock rate 800MHz
- The bus make the speed possible, and defines impedances, clocking and signals precisely.
- No control from RAS, CAS, R/W and CE signals
- Consists of a controller and a number of RDRAM modules connected together via a common bus, the bus is terminated at one end, and so NO RIMM can be left empty.
- Direct RAMBus, 16-bit-wide serial data path
- Capacity is limited by the max 32 RDRAM chips

RDRAM



RDRAM



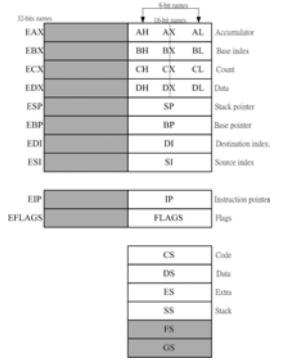
DRAM packages

- Single in-line memory module SIMM
  - Early 1990s, 30-pin, 72-pin, 32-bit data paths for 32-bit CPU
  - Pentium, 64-bits, memory banks, bank of memory as a logical unit
- Dual in-line memory module DIMM
  - Replace SIMM as standard, 168 pins in dual rows
  - Additional pins for CPU to retrieve DIMM information
  - 64 bits at a time instead of 32 or 16 bits
  - 3.3V DIMM emerged as the favoured standard
- Rambus in-line memory modules RIMM
  - Similar form factor as DIMM, 184 pins as compared to DIMM's 168
  - BIOS is able to determine the type of RAMs
  - System cannot use RIMMs unless BIOS and chipset support

Other DRAM features

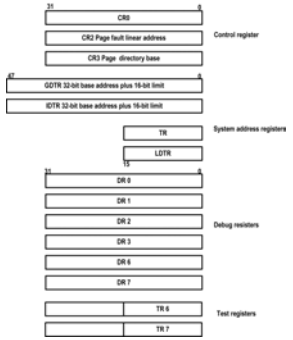
- **Presence detect**
  - Computer detects memory configuration when boot up
  - Parallel Presence Detect (PPD) used by SIMMs and some DIMMs
  - Serial Presence Detect (SPD) for SRAM, a special EEPROM chip stores information: size, speed, voltage, drive strength, row and column configuration
- **Parity memory**
  - Non-parity and parity checking uses a ninth memory chip to hold checksum data on the contents of other eight chips in the bank
  - If checksum does not match with actuall value, non-maskable interrupt (NMI) generated to instruction system to shut down for avoiding potential data corruption.
- **Error Check Code (ECC) memory**
  - Unlike parity memory, ECC uses more ECC bits for protection
  - Additional code required for ECC checking, overhead required for storing data, cause about 3% performance loss
  - It can automatically correct a single bit of errors

80X86 and Pentium Programming Model



EAX accumulator	Uses for instructions such as multiplication, division, and the adjustment instructions
EBX base index	Holds offset address of a memory location
EXC Count	Holds the counts for various instruction
EDX Data	holds result from multiplication or division
EBP base pointer	Points to memory location for data transfer
EDI destination index	Addresses string destination data
ESI Source index	Addresses string source data
EIP Instruction pointer	Addresses the next instruction
ESP Stack pointer	Addresses stack memory area
EFlAGS	Store condition and control status

80X86 and Pentium Programming Model



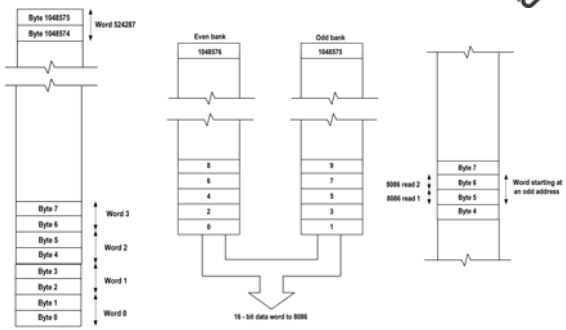
Control registers	Not in 8086/8088, to control processor in protected mode and testing
System address registers	Hold descriptor tables used in Protected Mode
Debug registers	Set breakpoints for debugging programs
Test registers	Test memory in the Translation Lookaside Buffer

80X86 and Pentium Programming Model

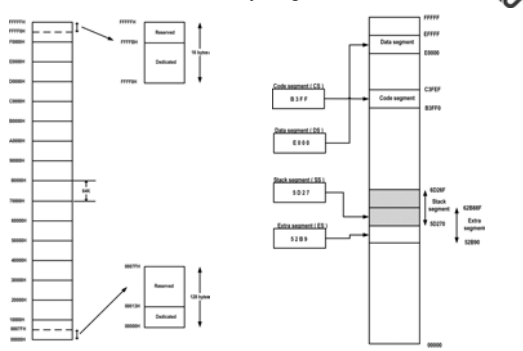
Note: The blank bits in the flag register are reserved for future use and must not be defined

Flag	Function	Flag	Function
C	Carry	IOPL	I/O privilege level
P	Parity	NT	Nested task
A	auxiliary	RF	resume
Z	Zero	VM	Virtual mode
S	sign	AC	Alignment check
T	trap	VIF	Virtual interrupt flag
I	Interrupt	VIP	Virtual interrupt pending
D	Direction	ID	identification
O	Overflow		

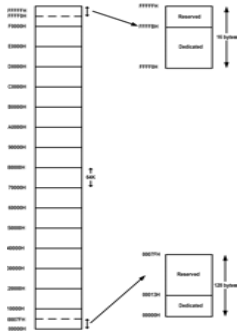
80X86 memory Organization



80X86 Memory Organization



80X86 Memory Organization



- 8086 Memory map
  - 64k byte block
  - Reserved and dedicated memory

Generating a 32-bit Address

80X86 Operating Modes

- Real Mode
  - Address space limited to 1MB, A0—A19, A20– are inactive
  - Segmented memory addressing , 64KB segment limit
- Protected Mode
  - Segment limit is 4GB
  - Pointers and descriptor table for segment addressing
  - Support virtual memory, segmentation and paging
  - Assign a privilege level to individual tasks
- New Protected Mode ---Virtual 8086 mode
  - Each Real Mode Program owns 1MB “chunk” of memory
  - Multiple program runs simultaneously but protected each other

Protected Mode (segmentation)

80286 descriptor

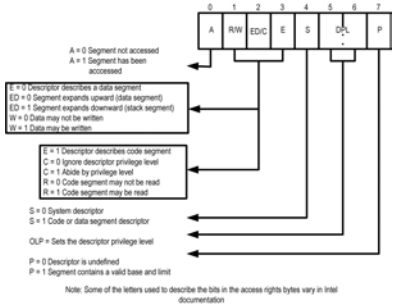
7	00000000	00000000	6
5	Access rights	Base(B23-B15)	4
3	Base(B15-B0)		2
1	Limi1(L15-L0)		0

80386 through Pentium 4 descriptor

7	Base(B31-B24)	G	D	O	A V	Limi1(L19-L16)		6
5	Access rights				Base(B23-B16)			4
3	Base(B15-B0)							2
1	Limi1(L15-L0)							0

- Segment descriptor format
- Global Descriptor for all programs, or call system descriptor
- Local Descriptor is unique to an application , call application descriptor
- Each descriptor table contains 8192 descriptors or totally 16,384 descriptors

Protected Mode (segmentation, access right)



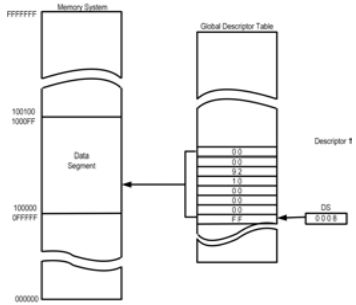
Protected Mode (segmentation, segment register)

RPL = Requested privilege level where 00 is the highest and 11 is the lowest

TI = 0 Global descriptor table
TI = 1 Local descriptor table

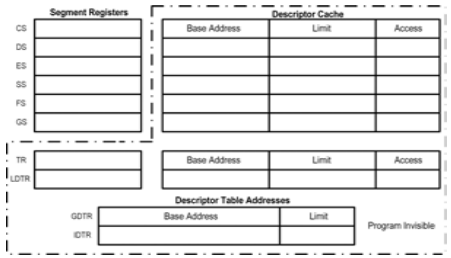
Selects one descriptor from 8192 descriptors in either the global or the local descriptor table

Protected Mode (segmentation)



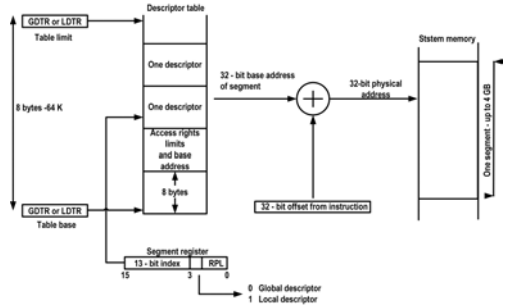
Using DS register to select a descriptor from global descriptor table

Protected Mode (program-invisible registers)

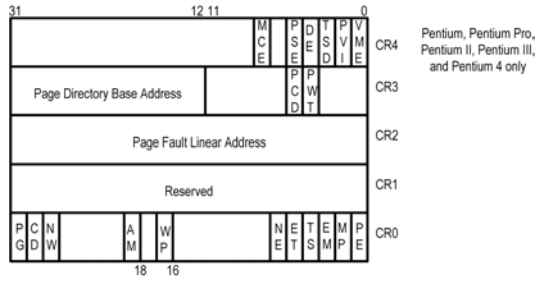


- A Cache memory internal to the processor
- Whenever a new segment number is placed in a segment register, descriptor table and descriptors are loaded into the cache until the register value changed

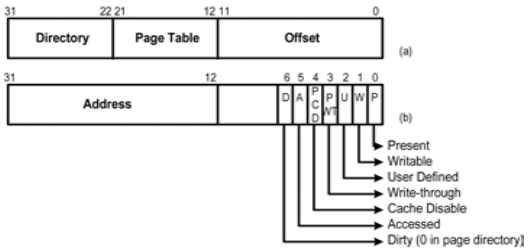
Protected Mode (segmentation)



Memory Paging (Paging Registers)

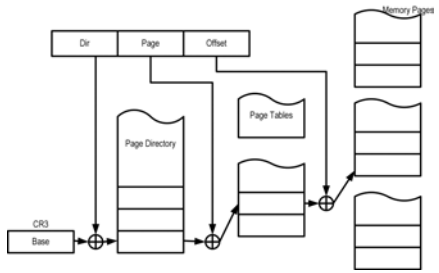


Protected Mode (paging registers)

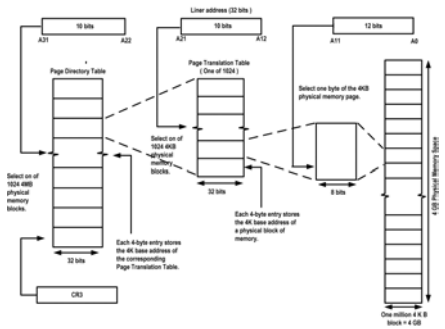


- Format for linear address
- Page directory or page table entry (CR3)

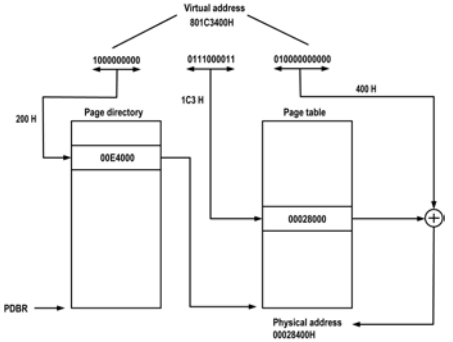
Protected Mode Addressing



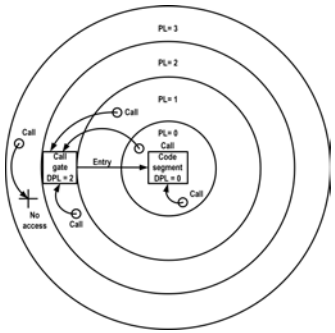
Protected Mode (paging)



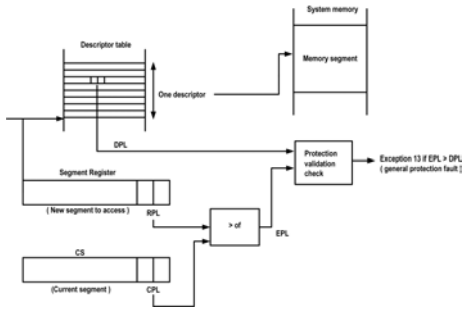
Linear to Physical Address Translation



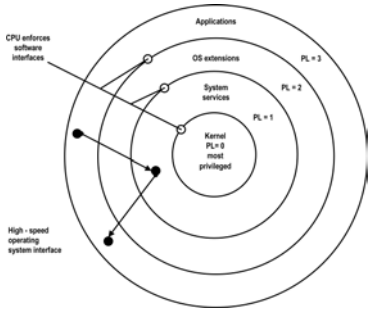
Virtual 8086 Mode



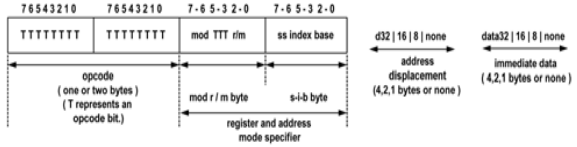
Virtual 8086 Mode



Virtual 8086 Mode

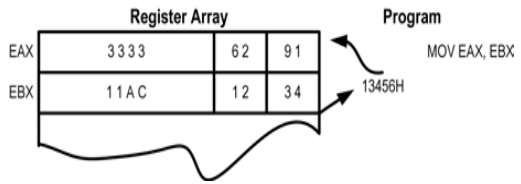


80x86 and Pentium Instruction Format

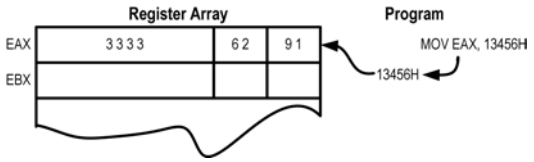


Intel Processor Addressing Mode

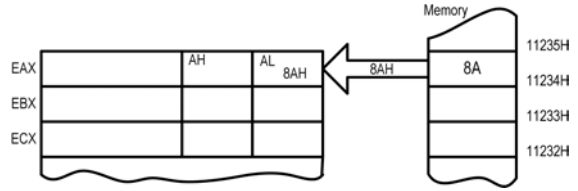
Register Addressing



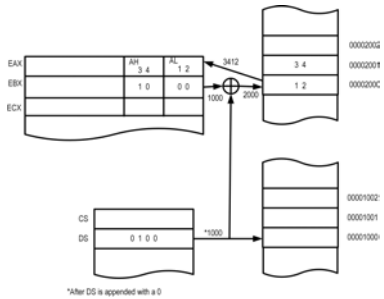
Immediate Addressing



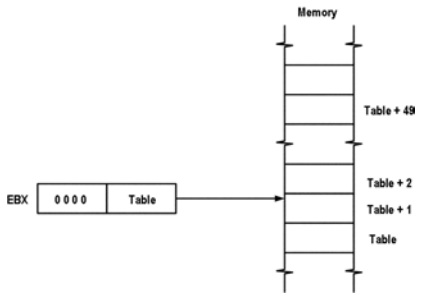
Direct Data Addressing



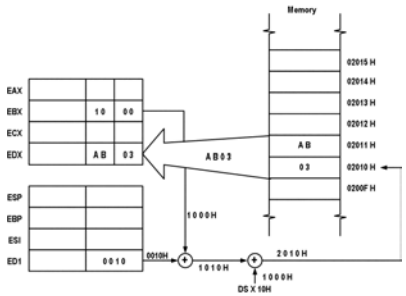
Register Indirect Addressing



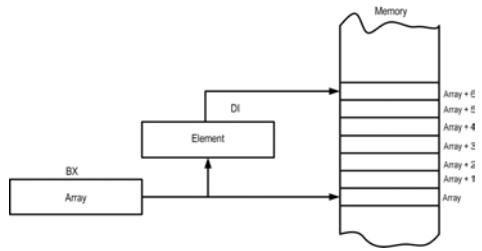
Register Indirect Addressing



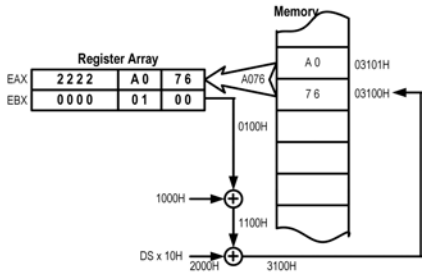
Base-Plus-Index Addressing



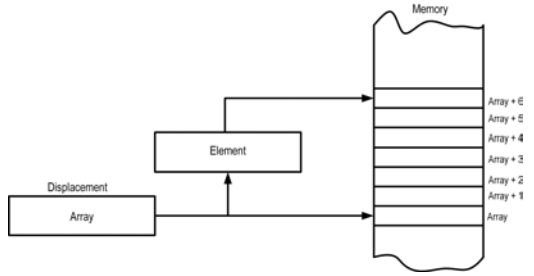
Base-Plus-Index Addressing



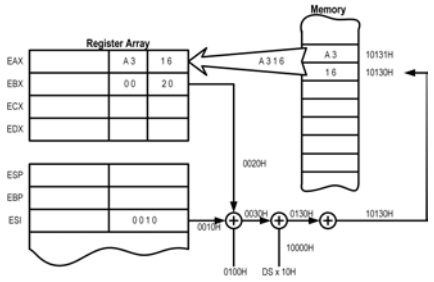
Register Relative Addressing



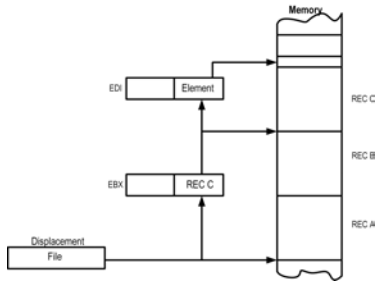
Register Relative Addressing



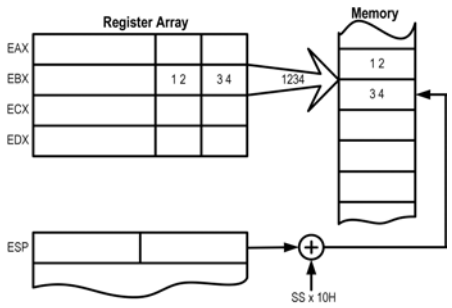
Base Relative-Plus-Index Addressing



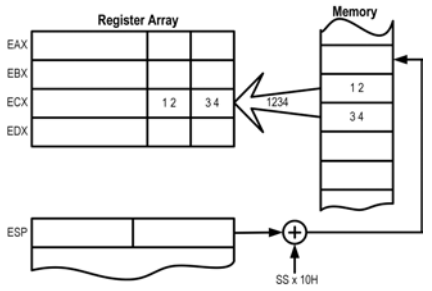
Base Relative-Plus-Index Addressing



Stack Memory-Addressing (PUSH)



Stack Memory-Addressing (POP)



Pentium Instruction Types

- Seven different instruction groups:
  - Data transfer
  - Strings
  - Arithmetic
  - Bit manipulation
  - Program transfer
  - Processor control

Pentium Instruction Types

- Data Transfer Instruction, e.g.
  - IN Input byte or word from port
  - LEA load effective address
  - MOV move to/from register/memory
  - OUT output byte or word to port
  - XCHG exchange byte or word
  - POPA pop all registers
  - PUSHA push all register
  - BSWAP byte swap

Pentium Instruction Types

- Arithmetic Instructions, e.g.
  - AAA ASCII adjust for addition
  - ADC add byte or word plus carry
  - CMP compare byte or word
  - DIV divide byte or word
  - IMUL integer multiply by byte or word
  - NEG negate byte or word
  - SUB subtract byte or word

Pentium Instruction Types

- Bit Manipulation Instructions, e.g.
  - AND logical AND of byte or word
  - NOT logical NOT of byte or word
  - OR logical OR of byte or word
  - RCL rotate left through carry byte or word
  - SAR arithmetic shift right byte or word
  - BTC bit test and complement
  - SETcc Set byte on condition

Pentium Instruction Types

- String Instructions, e.g.
  - CMPS compare byte or word string
  - LODS load byte or word string
  - MOVS move byte or word string
  - MOVSW move word string
  - REPE repeat while equal
  - STOS store byte or word

Pentium Instruction Types

- Program Transfer Instructions, e.g.
  - CALL call procedure
  - INT Interrupt
  - IRET return from interrupt
  - JC jump if carry set
  - JMP unconditional jump
  - LOOPE loop if equal
  - LEAVE leave a procedure

Pentium Instruction Types

- Processor Control Instructions, e.g.
  - CLC clear carry flag
  - CMC complement carry flag
  - STC set carry flag
  - STI set interrupt enable flag
  - LMSW load machine status word
  - LGDT load global descriptor table
  - CPUID CPU identification

Instruction Operation (Reference Site)

http://www.brookscole.com/compsci_d/templates/student_resources/0534953654_deckerhirchfield/aeonline/course/6/1/index.html