Sixth Semester B.E. Degree Examination, June/July–2013

Microprocessors

Time : 3 hrs. Max. Marks: 100

Notes: *Answer any* FIVE *full questions, selecting at least* TWO *questions from each part.*

PART - A

1. a. With a neat diagram explain the CPU architecture of 8086. (8M)

Ans: Please refer answer of Q.No. 1(a) of December–2011.

b. Define any four addressing modes used in 8086 microprocessor. Identify addressing modes used in each of the following 8086 instructions:

i) MOV, BX, 0354H

ii) ADD AL, [BX + 04]

iii) MOV AX, [BX + SI]

iv) MOV AX, [BX + SI + 04] (8M)

Ans: Please refer answer of Q.No. 1(a) of December–2011.

i) MOV, BX, 0354h	– Immediate addressing mode
ii) Add AL, [BX + 04]	– Memory indirect with based displacement(8bit)
iii) MOV AX, [BX + SI]	– Memory indirect with based, indexed
iv) MOV AX, [BX + SI + 04]	– Memory indirect with based indexed with 8 bit displacement

c. If DS = AB40H, CS = 9960H, SS = 3B00H, BP =7E74H, SP = 0135H, SI = 1245H, DI = 4356H, then determine physical address of the following instructions:

i) MOV [BP + DI + 6], AH

ii) ADD AL, [5036H] (4M)

Ans: **(i) MOV [BP + DI + 6], Ah**

Move Ah to memory pointed to by BP + DI + 06

PA = SR * 10h + offset

use SS since it is pointed by BP

SS * 10h + BP + DI + 06

3B00 × 10h + 7374 + 4356 + 06

= 3B000 h

7374

4356

06

466D0

PA = 466D0h

Move content of Ah to memory location at 466D0h pointed to by BP in SS

(ii) Add AL, [5036h]

It is by default DS

content at 5036h in DS is moved to AL

PA = DS × 10h + 5036h

= AB40h × 10h + 5C36h

AB400h

5036h

PA = B0436h

2. a. What do you mean by segment override prefix? Give an example. (4M)

Ans: MOV CS : [BX], AL

This instruction copies a byte from the AL register to a memory location. The effective address for the memory location is contained in the BX register. By default an effective address in BX will be added to the data segment (DS) to produce the physical memory address.

	2000 0 H	CS
+	002 0 H	BX
Phy. Addr.	2002 0 H	

In this instruction, the CS : in front of [BX] indicates that we want BIU to add the effective address to the code segment (CS) instead of DS by default to produce the physical address. The CS : is called **segment override** prefix.

Segment Override Prefix

The segment override prefix allows the programmer to deviate from the default segment. The segment override prefix is an additional 8-bit code which is put in memory before the code for the rest of the instruction. This additional code selects the alternate segment register. The code byte for the segment override prefix as the format 001XX110. The XX represents a 2 bits which are as follows : ES = 00, CS = 01, SS = 10 and DS = 11. It is important to note that the segment override prefix may be added to almost any instruction in any memory addressing mode.

b. Explain the role of AAD and AAM instruction of 8086 microprocessor with an example. (6M)

Ans: **AAD (ASCII adjust before division)**

Converts unpacked digits in AX to binary. It is done before division.

if AX = 0403 which is unpacked BCD for 43d

if cl = 07 and AX = 002Bh = 43d

now div cl ; divide Ax by cl ie 43/7 quotient = 6 → al

reminder = 01 → ah

AAM (ASCII adjust after multiplication) :

It is used to adjust product of to 2 packed BCD to unpacked digits in AX.

Function :

(a) Al = AL mod 10

(b) Ah = AL / 10

To multiply ASCII 8 with 6 say AL is stored with 8, BL is with 6

```
     8 × 6            0011    1000     = 38
     mul AL, BL       0011    .0110    = 36
                      0011    0000
```

AAM

```
Al = 48% 10 = 8        Ah      Al
Ah = 48/10 = 4         04      08
```

quotient in ah reminder in al (un packed digits of 48)

c. Write an assembly level language program to sort the numbers in ascending order using Bubble sorting technique. The program should be written using assembler Directives. (10M)

Ans: **ASCENDING ORDER USING BUBBLE SORT**

```
• MODEL SMALL                          ; Start of program
• DATA                                 ; Start of data segment
          A DB  05H,04H,03H,02H        ; Defining variables with values
          L DB  $ – A
• CODE                                 ; Start of code segment
          MOV  AX, @ DATA              ; Initialize the data segment
          MOV  DS, AX                    with the starting address
          LEA  DI, A
          MOV  BL, L                   ;
          DEC  BL                      ; Decrement BX reg by 1
PASS :    MOV CL, BL                   ; Contents of BX is moved to AX
          MOV  SI, 00H                 ; Clear the SI register
COMPARE:  MOV AL, A[SI]
          INC  SI INC SI               ; Increment SI by 1
          CMP  AL, A[SI]
          JB   NEXT                    ; If CF=1 jump to NEXT
          XCHG AL, A[SI]
          MOV  A[SI-1], AL
          DEC  CL
NEXT: .   JNZ  COMPARE
          DEC  BL                      ; Decrement BX register by 1
          JNZ  PASS                    ; If BX ≠ 0 jump to pass
```

```
          MOV  AH, 4CH                 ; Terminate the program
          INT  21H                     ; Function call
END                                    ; End of program
```

Output :

A = 02h, 03h, 04h, 05h

3. a. What are Assembler Directives? Explain the following directives with an example for each:

i) ASSUME

ii) PUBLIC and EXTRN

iii) GLOBAL

iv) ALIGN 16 (9M)

Ans: i) ASSUME: 8086 can address 4 physical segment at any time (i.e. CS,DS, SS, ES). The assume directive assigns logical segment to a physical segment at any given time.

eg. : Assume CS : Code, DS : data, SS : stack. Means for CS link Code, for DS link data area.

ii) PUBLIC and EXTRN:

PUBLIC : Large programs are usually written as several modules and each modules are individually assembled tested & debugged. When all modules are working correctly their object codes are linked to form a complete programs. To link the modules, any variable name or function or label referred to in other module must be declared public in the module where it is defined. The public directive in used to tell the assembler that a specified name or label will be accessed from other modules.

EXTRN : This directive in used to inform assembler that the names or labels following the directive are in some other module. The assembler will put information in the object code so that the linker can connect the two modules together.

Note : The names or labels referred to as an EXTRN in one module (calling prog.) must be declared PUBLIC in the module where it is defined.

```
eg. :   calling prog.                     called prog.
             • code
  EXTRN READKB : FAR                   PUBLIC READKB
                                        READKB PROC

          CALL READKB
               CODE                          RET
                                        READKB ENDP
```

iii) GLOBAL : The variable, constant or procedures declared global may be used by other modules also.

eg. : ROUTINE PROC GLOBAL

iv) **ALIGN 16 :** This directive makes the assembler align next data or instructions starting from first value specified in the directive. This alignment facilitates the access to word or double word data.

Format: Align value, value is power of 2 like 2, 4, 8, 16.

```
eg.: 0005h   Align 4
     0008h   double-wo  dd  0
```

This results double-wo variable alligned at an address divisible by 4.

b. Write an ALP to search a given character in the array of characters using string instructions. What is the role of SI, DI registers and DF bit? (5M)

Ans: Example on string instructions : ALP to find given sub-string is present or not in a main string. Result display FD if present. AB → absent, in the memory location.

```
• model small
• DATA
DStr. db "Do not seek too much advice" ; dest string do not end with 5
LD dw $-DStr
SStr db "ROCK"
LS dw $-SStr
Y db 2 dup (0)
• Code
    mov ax, @data
    mov DS, AX
    mov ES, AX          ; Initialise both DS, ES
    CLD
    mov dx, LD          ; length of dest string moved to dx
    LEA DI, DSTr.       ; DI points to dest string
LOCZ : LEA SI, SSTr.    ; SI points to source string
       mov CX, LS       ; length of source string to CX to search till CX=0
       PUSH DI          ; Store DI instack
       REPE CMPSB       ; Comp string byte & repeat till CX = 0 or ZF = 0
       JE LOCI          ; Now CX = 0 Source length is over & it is
       POP DI           ; found in dest. go to locl to display found
       INC DI           ;
       DEC DX           ; else take out DI pointer to dest string points
       JNZ LOCZ           to next byte reduce length by 1 & go to LOCZ if
                         length is non zero to continue search in the forward
                         director.
       mov y, 0ABh      ; If, destination string is over if DX = 0 it is
       mov ah, 4ch        not found terminate prog. by moving
```

```
       int 21h          AB to Y

LOCI : mov Y, 0FDh      ; FD TO Y for found
       mov ah, 4ch
       int 21h
       END
```

Role of SI → To store address and source string (array and character)

DI → To store address and destination string

DF → (Given character)

To move SI, DI in any direction.

Say Forward (DF = 0) reverse DF = 1.

c. Write an ALP to read a string from the keyboard and display the reversed string on the monitor screen. (6M)

Ans: Please refer answer of Q.No. 3(c) of December–2011.

4. a. Define interrupts. Explain TYPE0, TYPE1, TYPE2, TYPE3 and TYPE4 interrupts. (6M)

Ans: Please refer answer of Q.No. 4(b) of December–2011.

b. Explain hardware interrupts of 8086 microprocessor. (4M)

Ans: Please refer answer of Q.No. 4(c) of December–2011.

c. Differentiate macros and procedures. (4M)

Ans: Please refer answer of Q.No. 3(a) of June–2012

d. Write a macro to read a character without echo and to read a string of characters from the keyboard. (6M)

Ans: PROGRAM TO READ FROM KEYBOARD USNG MACROS MAC.ASM filename

```
REKB MACRO loc      ; rekb macro is defined with parameter loc
mov ah, 01h         ; reads a char from kb and store it in al
int 21h
mov loc, al         ; move the char read to loc from al
ENDM                ; end of macro definition
.model small
.stack 10h
.data
MEM db 2 dup (0)    ; initialise 2 mem data area
.code
start :
   mov ax, @data
```

```
        mov ds, ax
        REKB MEM          ; call the macro with actual argument mem
        REKB MEM+1        ; call the macro again with argument mem+1
        mov ah, 4ch       ; come out of the prog. to dos prompt.
        int 21h
        end start
```

NOTE: (1) For character read without echo MOV ah, 07h inplace of MOV ah, 01h

(2) For string replace MEM db 20 dup(0) to MEM db 20 dup(0)

PART - B

5. a. Define Stepper motor. Explain the interfacing of a stepper motor to 8086 microprocessor with necessary circuit diagram. Write an ALP to rotate the stepper motor clockwise by n steps and anticlockwise by m steps. (10M)

Ans: Please refer answer of Q.No. 5(b) of December–2011 and Q.No. 5(c) of December–2012

Here the count to be loaded in CX for rotating by an anlge θ is calculated as under :

Count = 360 / (Stepangle × No. of times we energise windings)

Convert this count to hexadecimal.

Also it is assumed that the windings are connected to port C. Hence PC acts as O/P port.

for anti-clockwise use data

077h, 0bbh, 0ddh, 0eeh in reverse order.

b. Interface 4x4 keyboard to 8086 microprocessor using 8255. Write the necessary circuit diagram and an ALP. (10M)

Ans: Please refer answer of Q.No. 5(b) of December–2012.

6. a. What are the functions of following 8087 instructions? Explain.

i) FCOMP

ii) FENI

iii) FDECSTP

iv) FSTENV

v) FYL2XPI (10M)

Ans: i) FCOMP :Compare and POP with implicit arguments. (same as FCOM except SP is updated)

FCOMP ; Compare St with St (1) and SP ← SP + 1

Here St (1) is source, St is destination

SP is incremented by 1 after comparison and old St (1) is new St.

FLD num1 ; St (1) ← num1

FLD num2 ; St ← num2

FCOMP ; num2 – num1

ii) **FENI :** Enables exception interrupts PIN.

iii) **FDECSTP :**

iv) **FSTENV :**Stores the environment of the coprocessor to dest (except stack registers) while calling procedures.

FSTENV dest : save control word, status word, tag register, pointers except stack register to memory.

This is used in procedure calls to save information related to coprocessor except stack.

v) **FYL2XPI :**This computes Y * $\log_2$ (x+1) SP ← SP + 1

Here X is at ST, Y is at ST (1)

Perform Y * $\log_2$ (x+1) and result is stored at St (1) ; SP ← SP + 1

Values of X and Y are in the range

$0 < x < \infty$ and $\infty < y < +\infty$

FLD Y ; St (1) ← Y

FLD X ; St (0) ← X

FYL2XPI ; St (1) ← Y * $\log_2$ (x+1)

b. Write a program using 8087 instructions to compute the volume of the sphere using MASM syntax. (6M)

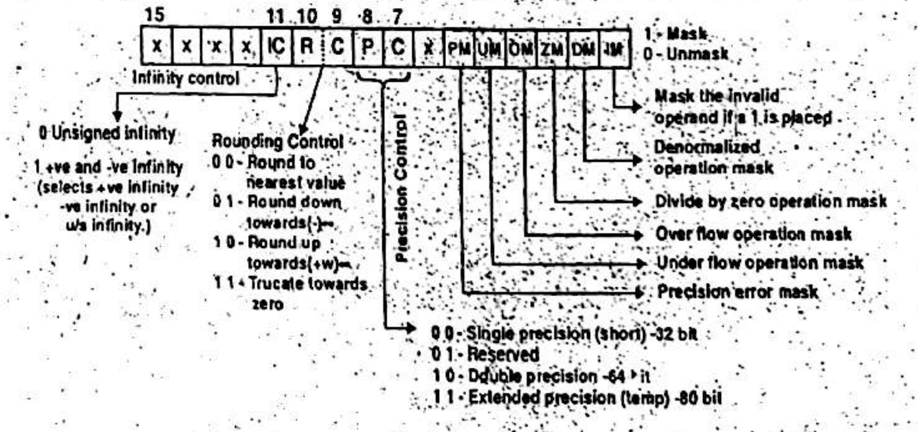
Ans: Please refer answer of Q.No. 6(a) of December–2011.

For sphere area Asphere = $\frac{4}{3}$ PIR^3 or Asphere = 0.75 × PI × R × R × R

use above program and use Fmul St, St Two times.

c. Explain the control register format of 8087. (4M)

Ans: It is used to select and control the status word registers like precision, rounding


15
11 10 9 8 7
X X X X IC R C P C X PM UM OM ZM DM IM
1 - Mask
0 - Unmask
Infinity control
0 Unsigned infinity
1 +ve and -ve Infinity (selects +ve infinity -ve infinity or u/s infinity.)
Rounding Control
0 0 - Round to nearest value
0 1 - Round down towards(-)∞
1 0 - Round up towards(+w)∞
1 1 - Trucate towards zero
Precision Control
0 0 - Single precision (short) -32 bit
0 1 - Reserved
1 0 - Double precision -64 bit
1 1 - Extended precision (temp) -80 bit
Mask the invalid operand if a 1 is placed
Denocmalized operation mask
Divide by zero operation mask
Over flow operation mask
Under flow operation mask
Precision error mask

control, infinity control and masks and unmasks the error (exception) bits of status word register.

FLDCW instruction is used to load value to CW register.

7. a. With a neat diagram explain the maximum mode operation of 8086. (8M)

Ans: Please refer answer of Q.No. 7(a) of June–2012.

b. What are the characteristics of PCI and USB interface? (6M)

Ans: Please refer answer of Q.No. 7(b) of December–2012.

c. Interface Printer 8086 processor with relevant signals of importance. Explain using a flowchart. (6M)

Ans: The Parallel Printer Interface (LPT)

The parallel printer interface (LPT) is located on the rear panel of the personal computer. The LPT stands for the printer. This printer interface gives eight data lines to transfer data alongwith the control signal usually called hand shaking signals to control the flow of data. The printer interface can be programmed to receive or send data. This allows devices rather than printer, such as CD-ROMs, to be connected to and used by the PC through the parallel port. The centronics protocol is a printer protocol which specifies the standards for printer interface.

Interface Details

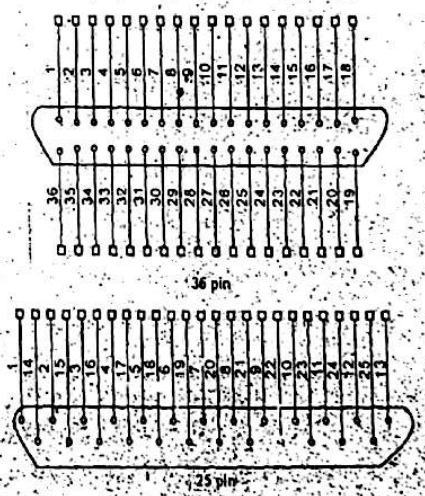
The computer can have two parallel printer interface, commonly known as parallel ports (LPT1 and LPT2). The LPT1 is normally at I/O port addresses 378H, 379H and 37AH. The LPT2 port, if present, is located at I/O port addresses 278H, 279H and 27AH. Let us see the details of both ports, but we use LPT1 port addresses. The Table shows the pins and signals for parallel/centronics interface and the Figure shows the connectors used for parallel/centronics interface.

Parallel centronics port pins and signals

Signal	Description	Pin no. for 25 pin connector	Pin no. for 36 pin connector
$\overline{STR}$	Strobe to printer	1	1
D0	Data bit 0	2	2
D1	Data bit 1	3	3
D2	Data bit 2	4	4
D3	Data bit 3	5	5
D4	Data bit 4	6	6
D5	Data bit 5	7	7
D6	Data bit 6	8	8
D7	Data bit 7	9	9
$\overline{ACK}$	Acknowledge from printer	10	10
BUSY	Busy from printer	11	11
PAPER	Out of paper	12	12
ONLINE	Printer is online	13	13
$\overline{ALF}$	Low if printer issues a LF after a CR	14	14
$\overline{ERROR}$	Printer error	15	15
$\overline{RESET}$	Resets the printer	16	31
$\overline{SEL}$	Selects the printer	17	36
+5V	5V from printer	–	18
Protective ground	Earth ground	–	17
Signal ground	Signal ground	All other pins	All other pins

NOTE: Bar indicates an active low signal.

Connectors used for parallel/centronics port



Control/Handshaking Signals for Printer Interface

a) **Input signals for printer**

1. $\overline{INIT}$: This signal when activated tells the printer to perform its internal initialization sequence.
2. STROBE ($\overline{STB}$) : This signal when activated tells the printer that valid data is available on the data bus.

b) Status signals output from printer

1. $\overline{\text{ACK}}$: This signal when low indicates that the data character has been accepted and the printer is ready for the next data.

2. BUSY : This is active high signal. It goes high when printer is not ready to receive a character.

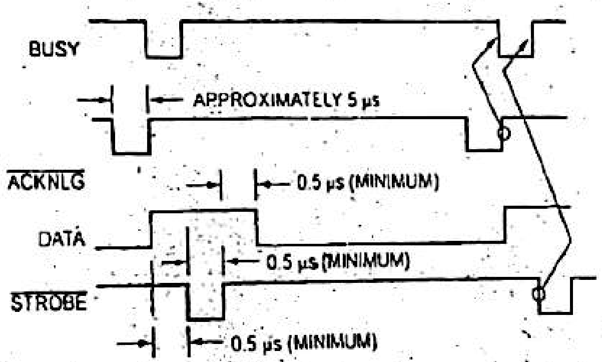
3. PE : This active high signal goes high when printer is out of paper.

4. SLCT : This signal goes high if the printer is selected for receiving data.

5. ERROR : This active low signal goes low for variety of problem conditions in the printer.

Figure shows the timing waveforms for transfer of data characters to an IBM printe using the basic handshake signals.

Timing waveforms for transfer of data characters to an IBM printe



Computer sends the INIT pulse for at least 50 μs, to initialize the printer. Computer then checks for BUSY low to confirm whether the printer is ready to receive data or not. If BUSY signal is low (not busy), computer sends an ASCII code on eight parallel data lines and after at least 0.5 μs, it also sends $\overline{\text{STB}}$ signal to indicate valid data is available on the data bus. Computer activates this $\overline{\text{STB}}$ signal for at least 0.5 μs and it also ensures that valid data is present on the data bus for at least 0.5 μs after the $\overline{\text{STB}}$ signal is disabled. When the printer is ready to receive the next character, it asserts its $\overline{\text{ACK}}$ signal low for about 5 μs. The rising edge of the $\overline{\text{ACK}}$ signal tells the computer that it can send the next character. The rising edge of the $\overline{\text{ACK}}$ signal also resets the BUSY signal from the printer. When computer finds busy low, it sends the next character along with strobe and the sequence is repeated till the last character transfer.

The system program written to carry out data transfer through parallel port uses data port (378H), the status register (379H) and an additional status port (374H). The bits patterns for these ports are as shown in Figure.

8. Write short notes for the following:

a. 80386 special registers (6M)

b. Salient features of 80486 processor (6M)

c. Pentium CPU architecture (8M)

Ans: (a) 80386 special registers:

80386 has the following registers.

(1) 4-General purpose registers - 32 bit - EAX, EBX, ECX, EDX

2 pointers - ESP, EBP, 2 index registers – ESI, EDI can be divided as 32, 16, 8 bit (EAX, AX, Ah or Al)

(2) Six segment registers - with 16 bit capacity

CS, DS, SS, ES, FS, GS (ES, FS, GS are extra segments)

(3) Index, pointers & base registers (EIP, BP, SP, SI, DI & BX) of 16 bit except EIP - it is 32 bit.

(4) Flag register - 32 bit only 13 flags are defined 6 condition flags, 4 system flags, 3 control flag.

(5) 4-system address registers - GDTR, IDTR (48 bit) LDTR, TR (16 bit) (Task selector register) (GDTR – Global Descriptor Table Register, IDTR – Interrupt Descriptor Table Register)

(6) 4 control registers - CR0 - CR3 (32 bit)

(7) 6 Debug register - DR0 - DR7 (32 bit capacity)

DR4, DR5 undefined only DR6 - DR7 is show here are defined

(8) 2 Test registers (TR0 - TR7) only TR6, TR7 are defined

(b) Salient features of 80486 processor:

1) It is a 32 bit with on-chip memory management and cache memory units with 168 PIN grid array, works with 25 MHZ, 33 MHZ, 50 MHZ and 100 MHZ.

(2) It has built in math coprocessor which avoids sending signals from microprocessor to coprocessor for any operation saves (15 cp).

(3) MMU provides 4 levels of protection for isolating and protecting applications and the O.S. from each other.

(4) It operates in 3 modes – Real, Protected and Virtual 8086 mode with one extra flag compared with 80386 that is Allighment check AC flag.

(5) Most of 80486 instructions require only one clock instead of 2 clock required by 80386.

(6) It supports 5 stage instruction pipeline scheme hence faster than 80386. Two out of 5 stage are used for decoding the complex instructions and balance 3, one each for storing decoding execution.

(7) It has few new instructions that controls the internal cache. It performs XADD and CMPXCHG with an exchange and swap of byte (B) BSWAP.

Six new instructions:

INVD - invaidate the cache

WBINVD - Write back operation

INVLPG - Invalidate the TLB entry

BSWAP - Swap the order of bytes in 32 bit register.

XADD - Add and exchange.

CMPXCHG - compares dest with accumulator, if accumulator and destination are different the accumulator is replaced by destination.

(8) It supports built-in-self-test. It test microprocessor, coprocessor, and cache at reset time and sets EAX to zero. Also it has on-line parity check.

(9) It has additionai test registers TR3 - TR5 to test cache memory.

(c) Pentium CPU architecture:

Introduction to pentium microprocessor

Pentium signals an improvement to 80486 with respect to cache, databus, faster numeric coprocessor, dual integer processor and branch prediction logic. The cache has been reorganized to form two caches each of 8KB one for data and other for instructions.

The data bus is 64 bits. The numeric coprocessor operates at five times faster than 80486 numeric coprocessor. The dual-integer procesor allows two instructions per clock. The branch prediction logic allows programs that branch to execute more efficiently. All these changes are internal to pentium and also pentium has added feature of MMX instructions. (Multimedia instructions)

The pentium pro is still faster version of pentium and contains a modified internal architecture that can schedule upto 5 instructions for execution and even faster floating point unit.

Pentium pro also contains a 256KB or 512KB level-Two (L-2) cache in addition to 16KB (8KB for data and 8KB for instruction) Level one cache (L1).

It has additional 4 address lines and can access 64GB memory space.

Pentium Salient Features

(1) It has wider data bus. Its data bus is 64 bit it supports burst read and write cycles. Also bus cycle pipelining has been added to allow 2 bus cycle to be in progress simultaneously.

(2) Faster FPU : It provides upto 10 times speed-up for applications like add, mul and load.

(3) Improved Cache : It has seperate cache for code and data on the same chip each of 8Kb with 32 byte line size. Each cache has a dedicated Translation lookaside buffer (TLB) to translate linear addresses to physical addresses.

(4) Dual integer processor allows execution of 2 instructions per clock.

(5) It has branch prediction logic to check whether a branch will be valid or invalid with 2 prefetch buffer one for code in linear fashion and the other for branch target buffer.

(6) It has data parity check (DP_0 - DP_7) and address parity checking.

(7) It has functional checking with a second processor the "checker" executes in lock step with master and compares the masters output and asserts an error signal if a mismatch occurs.

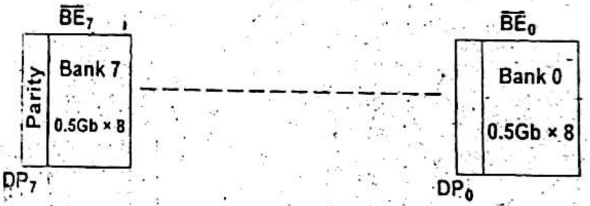
(8) Superscaler processor facility makes parallel instruction execution of multiple instructions that is 2 integer or floating point instructions simultaneously execution facility.

Memory System

It uses 64 bit databus and 8 banks with each of 0.5 Gb. The address bus is 32 bit and hence memory capacity is 4Gb (00000000 TO FFFFFFFFh). Each bank can store a byte of data with parity check bit and are selected by $\overline{BE_0}$ To $\overline{BE_7}$ lines. The selection is either a single byte, word, double word or quad word.

It can retrieve double precision (64 bit) floating point number in one read cycle since data bus is 64 bit.

It can also check and generate parity for address bus (A_5 - A_{31}) with $\overline{APCHK}$ signal.



Special Pentium Registers

All registers of 80386/486 are available in pentium with differences in CR_0 (control register) Flag register and machine check type register and also addition of CR4 control register.

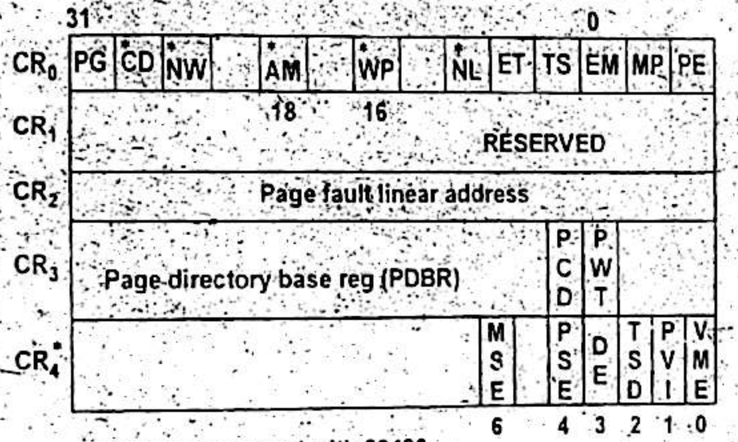
EFLAG : It has 17 flags (with 3 extra flags over 486)

	ID	VIP	VIF	AC	VM	RF	NT	IOPL	0F	DF	IF	TF	S	Z	AF	P	-	CF

with 32 bit capacity

CONTROL REGISTER

VIP - Virtual interrupt pending
VIF - Virtual interrupt is the image of virtual interrupt flag used with VIP
ID - Identification flag for CPUID instruction



•- change compared with 80486

CD - Cache disable CD = 1 cache will not fill.

NW -Not write through NW = 1 cache is disabled for write.

AM -Alignment mask enables to occur only for protected mode when it is set.

WP - Write protect to protect user level pages against supervisor level write operation when WP = 1 supervisor can write to user level.

NE - Numeric error for coprocessor error detection if NE = 1, $\overline{FERR}$ pin is active else coprocessor error is ignored.

VME-Virtual mode extension for virtual interrupt flag if VME = 0 virtual interrupt support is disabled.

PVI -Protected mode virtual interrupt to support virtual interrupt flag in protected mode.

TSD- Time stamp disable to control RDTSC instruction (Read time stamp counter)

DE - Debugging extension Enables I/O break point debugging extension when set.

PSE- Page size extension - To enable 4Mb memory pages when set.

MCE-Machine check enables - Enables the M/C checking interrupt.

-Machine check type register (MCTR) - 64 bit to store copy of address and control signal when parity error is detected.

New Pentium Instructions

(1) CMPXCHG8B -Compare and exchange 8 bytes. It compares 64 bit number stored in EDX, EAX with 64 bit memory or a pair of registers, if equal the value in CS:EBX is stored into specified memory operand and zero flag is set, else The contents of memory operand is copied into EDX : EAX.

(2) CPUID - To read CPU identification code and other information from the pentium.

(3) RDTSC - Reads the current value of internal 64 bit time stamp counter into EDX : EAX that is updated every clock cycle.

(4) RDMSR/WRMSR - Used to access the contents of selected 64 bit model specific register (MSR) specified by ECX and are deposited into pair EDX : EAX for reading and for writing EDX : EAX to target specified in MSR.

If ECX = 0 TO MCAR

If ECX = 1 TO MCTR

If ECX = 0Eh TO Test register

(5) RSM - Returns from system management interrupt.

•••••