Assignment No. 5

D D M M Y Y Y Y

Q 1. Represent the following real numbers in 32 bit single and double precision floating point number representation.

(a) +7.5

→ (1) single precision floating point format.

sign bit (s) = 0

$7.5 = 0111.1 \times 2^0$

$= 1.111 \times 2^2$

- Exponent is excess -127 in single precision format

Bias Exponent = Actual Exponent + Bias Value

= 127 + 2

= 129

Mantissa (m) = (0.111)

0 10000001 11100000000000000000000

∴ +7.5 = 0x40F00000

(2) Double precision floating point format

sign bit (s) = 0

$7.5 = 0111.1 \times 2^0$

$= 1.111 \times 2^2$

- Exponent is excess - 1023 in double precision format

Bias exponent = Actual exponent + Bias value

= 1023 + 2

= 1025

Mantissa (M) = (0.111)

0 10000000001 11100000000000000000000000000000000000000000000000

∴ +7.5 = 0x401E 0000 0000 0000

(b) -24.75

→ (1) single precision floating point format

sign bit(s) = 1

$24.75 = 11000.11 \times 2^0$

$= 1.100011 \times 2^4$

- Exponent is excess - 127 is single precision format

bias exponent = actual exponent + bias value

= 127 + 4

= 131

Mantissa(m) = (1.100011)

1 10000011 10001100000000000000000

∴ -24.75 = 0xc1c60000

(2) Double precision floating point format

~~sing~~bit sign bit(s) = 1

$24.75 = 11000.11 \times 2^0$

$= 1.100011 \times 2^4$

- Exponent is excess -1023 in double precision format bias exponent = actual exponent + bias value

= 1023 + 4

= 1027

Mantissa (m) = (1.100011)

1 10000000011 10001100000000000000000000000000000000000000000000

∴ -24.75 = 0xc038 c000 0000 0000

Q2. List different ~~types of~~ kinds of floating point numbers in IAS2 architectures. Also draw formats.

→ These are three kinds of floating point numbers in IAS2 architecture.

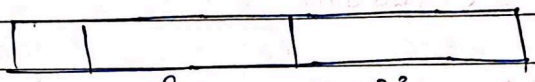
D D M M Y Y Y Y

* IEE754 single precision floating point format

→ There are 32 bit wide numbers and have three parts - a sign, a mantissa, and an exponent.

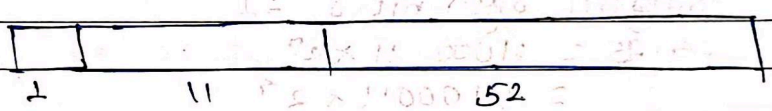
→ Sign is 1 bit wide and represents the sign of the number. Mantissa is 23 bit wide and represents the fractional part of the floating point number in binary. A leading one bit integer is not stored and is assumed as 1 in the normalized numbers.

→ Exponents are stored in excess-127 representation.



* IEEE754 Double precision floating point format

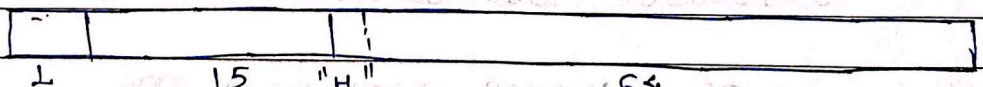
→ These are 64 bit wide with sign, mantissa and exponent in excess 1023 representation



* Double extended precision Floating point numbers

→ Double extended precision floating point nos. are 80 bit wide

→ This no format is used to perform computation internally in IA32 processors when instruction in x87 Floating pt. instruction set are used.



Q5 Write short note on.

(a) x87 data registers in IA32 processor

→ It include 8 general purpose data registers, each 80 bit wide and capable of storing a real no. in double extended precision format

→ As the data is loaded into the FPU stack, the stack top moves downloadward in the register.

→ When eight value have been loaded into the stack, all eight ~~value~~ FPU data registers have been utilized.

→ If ninth value is loaded into the stack, the stack pointer wraps around to the first register and replaces the value in that register with the new value.

→ When an integer, BCD, single, double precision operands is loaded into one of these data register, the operand is implicitly converted to double extented precision format.

→ In addition of data registers, x87 also includes three 16 bit registers which are used in controlling the computation

→ These registers are the following;

- x87 control register
- x87 status register
- x87 tag register

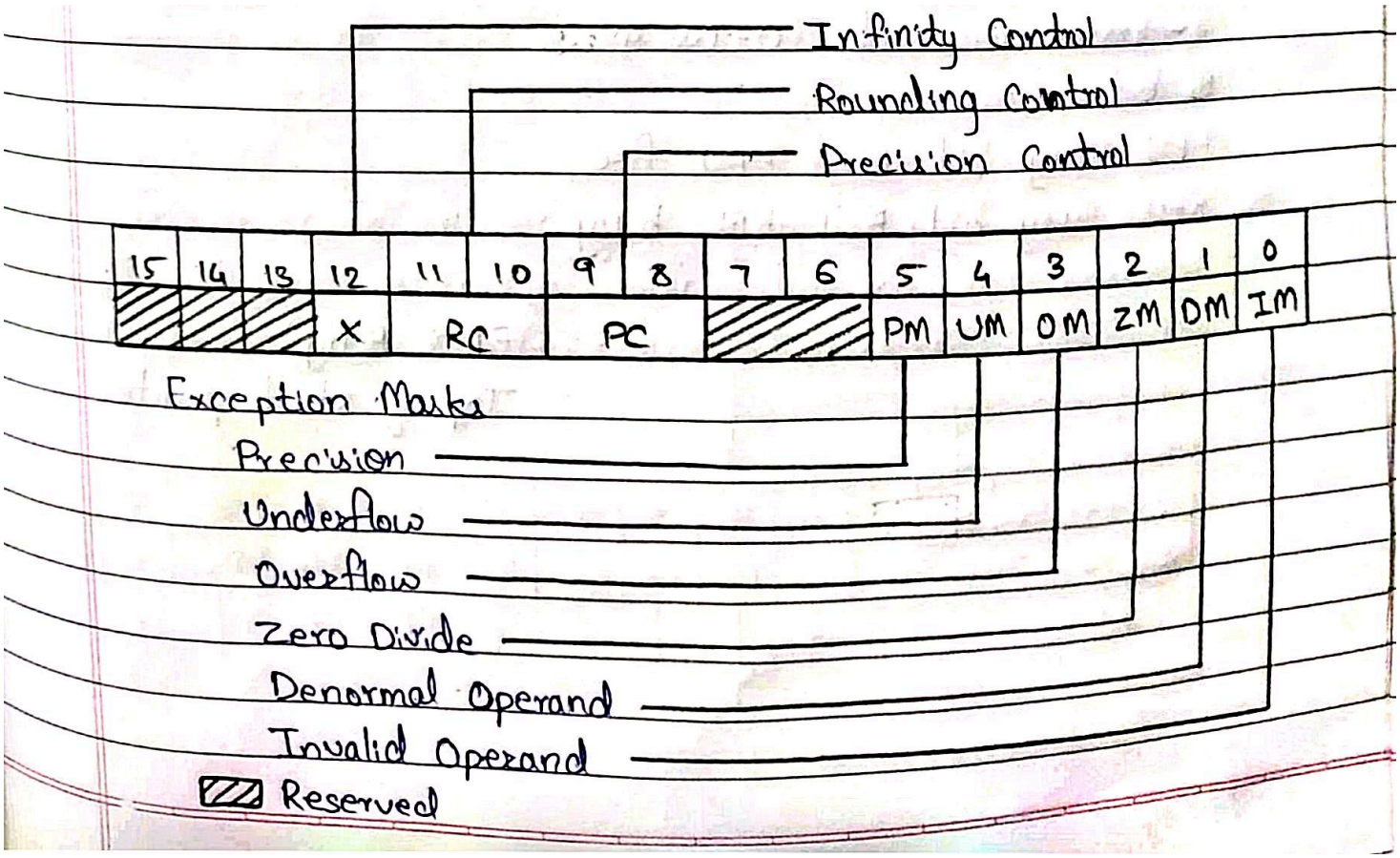
(b) x87 control register.

→ - During computation, several floating point error condition such as divide by zero occur

- In case of such errors, x87 FPU can be performed to raise floating point execution.
- Bit 0 to Bit 5
  0 indicate → unmask exception.
  1 indicate → mask exception.
- used to handle the precision and rounding of operands
- Infinity bit always set to 0.

D D M M Y Y Y Y

* Precision Control	* Rounding control
00 - single precision	00 - Round to nearest
01 - unused	01 - Round ~~to~~ down
10 - double precision	10 - Round up.
11 - double extended precision	11 - Round towards zero


Infinity Control
Rounding Control
Precision Control
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
X RC PC PM UM OM ZM DM IM
Exception Masks
Precision
Underflow
Overflow
Zero Divide
Denormal Operand
Invalid Operand
Reserved

c. X87 Status Register

→ 1. Top: It is 3 bit field contain index to register assumed to be at the top of the stack.

2. Four conditional code flags C0 to C3

A floating pt. no. can be compared in different ways with other floating pt. no.

The result of comparison is stored in either eflags or in floating pt. conditional code C0 - C3 ~~underflow conditions~~

3. SF: The stack fault bit in ~~either~~ status register indicates error due to stack overflow or underflow conditions.

Example: When one data on stack & addition operat^n is performed which takes two data from the stack, a stack fault occurs.

Stack overflow: when all 8 registers are in use & an attempt is made to load an data, stack overflow occurs.

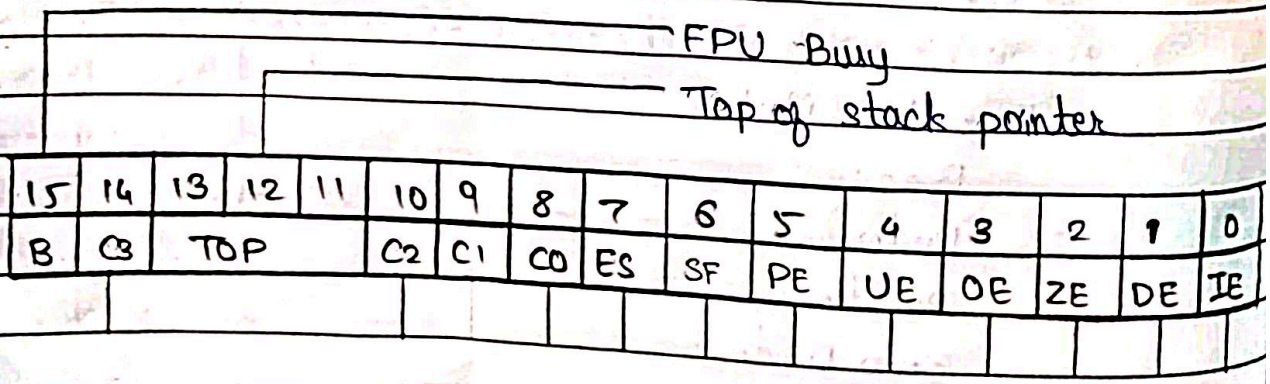
ES - Error Status

1 - Error occured 0 - No error

B bit

FPU Busy bit B=0 FPU free

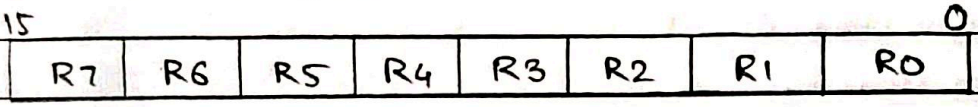
FPU Busy bit B=1 FPU busy



d. x87 Tag register

→ - Each data register of x87 FPU has an associated 2 bit tag contained in a tag register.
- The code in the tag register indicates the type of the value of the corresponding data register.
- Tag is associated with data register.

00 - R has a valid value.
01 - R has a zero.
10 - R has a special value (Nan, +/- infinity, renormalized no.).
11 - R is empty.



Q4. Explain the following arithmetic instructions.

(a) fadd

→ Add two floating point no. available in stack registers.
Eg: fadd %.st(1), %.st(0)
Here, %.st(0) is added with %.st(1) and result restored in destination stack %.st(0)

(b) fiadd

→ It is used to add an integer stored in memory location to the floating point no. on the top of stack.

(c) fsubrp

→ This instruction perform the additional operation of popping the FPU register stack following the subtraction. to pop the register stack, the processor marks the st(0) register as empty and increments the stack pointer (TOP) by 1..

D D M M Y Y Y Y

(d) fidivr

→ This instruction converts an integer source operand to double extended-precision floating point format before performing the division

(e) fimul

→ This instruction converts an integer source operand to double extended-precision floating pt format before performing the multiplication.

(f) fchs

→ This instruction is used to change the sign of the input arguments on the top of stack.

Q5: List & explain any four each constant loading & Trigonometric x87 FPU instructions.

→ - In x87 instruction set, there are certain instructions that can load commonly used constants on the stack top. These instructions are the following.

Instructions	Constant pushed in ST
fld1	st(0) = 1.0
fldz	st(0) = 0.0
fldpi	st(0) = 3.14
fldl2e	$\log_2 e$
fldln2	$\log_e 2$
fldl2t	$\log_2 10$
fldlg2	$\log_2 10$

• Trignometric x87 FPU instructions

- In the x87 FPU instruction set, following instructn are availble & can compute trigonometric functions for arguments stored on the stack top.

- Fsin : It compute sin of angle in radians provided in st(0).
- Fcos : It compute cosine of angle in radians provided in st(0).
- Fsincos : It compute & leaves two values in the reguster stack.
- Fptan : It removes the angle provided on top of the reguster stack & computees its tangent.

Q6. Write assembly language program in 80386 to compute following complex mathematical operation using the IA32 FPU instructions. ((43.65/22)+(76.34×3.1))/((12.43×6)÷(140.2/94.21))

→
```
.section .data
value1:
  .float 43.65
value2:
  .int 22
value3:
  .float 76.34
```

```
value4:
.float 3.1
value5:
.float 12.43
value6:
.int 6
value7:
.float 140.2
value8:
.float 94.21
.section .text
.globl _start
_start :nop
finit
flds value1
fdiv value2
flds value3
flds value4
fmul %st(1), %st(0)
faddd %st(2), %st(0)
flds value5
fimul value6
flds value7
flds value8
fdivrp
fsubr %st(1), %st(0)
fdivr %st(2), %st(0)
movl $1, %eax
movl $0, %ebx
int $0x80
```

Q7. Write assembly language program in 80386 to find roots of quadratic equation using IA32 FPU instructions.

→
```
.section .text
.globl _start
_start : nop
    flds a
    fimul val1
    flds c
    fmul %.st(1), %.st(0)
    flds b
    fmul %.st(0), %.st(0)
    fsubp %.st(1), %.st(0)
    fsqrt
    flds b
    fchs
    fadd %.st(1), %.st(0)
    fldsb
    fchs
    fsub %.st(2), %.st(0)
    flds a
    fimul val2
    fsts val5
    fdivr %.st(2), %.st(0)
    fsts val3
    flds val5
    fdivr %.st(2), %.st(0)
    fsts val4
movl $1, %.eax
movl $0, %.ebx
int  $0x80
```

Q8 Write assembly language program in 80386 to convert degree to radian & compute its sin cosine & tan value.

→

```
.section .data
degree1:
    .float 90.0
val180:
    .int 180
.section .bss
    .lcomm radian1, 4
    .lcomm result 1, 4
    .lcomm result 2, 4
    .lcomm results, 4
.section .text
.globl _start
_start: nop
finit
    flds degree1
    fidivs val180
    fldpi
    fmul %st(1), %st(0)
    fsts radian1
    fsin
    fsts result1
    flds radian1
    fcos
    fsts ~~root~~ result2
    flds radian1
    ~~f~~ftan
    fsts result3
    movl $1, %eax
    movl $0, %ebx
    int $0x80
```

Q7. Write assembly language program in 80386 to compute $\log_b x = (1/\log_2 b) \times \log_2 x$ where b=10, x=17

→
```
.section .data
value:
  .float 17.0
base:
  .float 10.0
.section .bss
  .lcomm result, 4
.section .text
.globl _start
_start: nop
finit
fld1
flds base
fyl2x
fld1
fdivp
flds value
fyl2x
fsts result
movl $1, %eax
movl $0, %ebx
int $0x80
```

Q10. Write a short note on SIMD Environment.

→ - SIMD technology provides additional way to define integer.
- It performs arithmetic operat^{n} on a group of multiple integers simultaneously.
- SIMD architecture uses packed data type.
- A packed integer is series of bytes that can represent more than one integer.
- SIMD instruction set provide a few register called MMX registers & XMM registers.
- MMX registers includes 8, 64 bit registers named mm0 - mm7.
- Data register are 80 bit wide, out of which only 64 bit are used by instructions in the SIMD instruction set.
- It is recommended to use finit instruction to initialize X87 FPU at the time of switching from SIMD to X87 environment.
- XMM registers support integer & floating pt. data types.
- XMM registers includes 8, 128 bit registers named xmm0 - xmm7.
- Supported floating pt. data type include a packed single precision floating pt. nos.

Q12. Write 80386 program to perform MMX addition on following integer value 10 & 20 with 30 & 40 nos.

→
```
.section .data
value1:
  .int 10,20
value2:
  .int 30,40
.section .bss
  .lcomm result,8
```

```
.section .text
.globl _start
_start: nop
    movq value1, %mm0
    movq value2, %mm1
    paddd %mm1, %mm0
    movq %mm0, result
  movl $1, %eax
  movl $0, %ebx
  int  $0x80
```

Q13 Write 8038s program to perform SEE addition on following four single precision floating pt. no.

value1: 12.34, 2345.5, -93.2, 10.44

value2: 39.234, 21.4, 100.94, 10.56

→
```
.section .data
.align 16
value1:
  .float 12.34, 2345.5, -93.2, 10.44
value2:
  .float 39.234, 21.4, 100.94, 10.56
.section .bss
  .lcomm result, 16
.section .text
.globl _start
_start: nop
    movaps value1, %xmm0
    movaps value2, %xmm1
    addps %xmm1, %xmm0
    movaps %xmm0, result
  movl $1, %eax
  movl $0, %ebx
  int  $0x80
```