ASSIGNMENT NO 13

AIM: - Write 8087 or X86/64 Assembly language program (ALP) to obtain:

- Mean
- Variance
- Standard Deviation

For a given set of data elements defined in data segment. Also display result.

APPARATUS:-

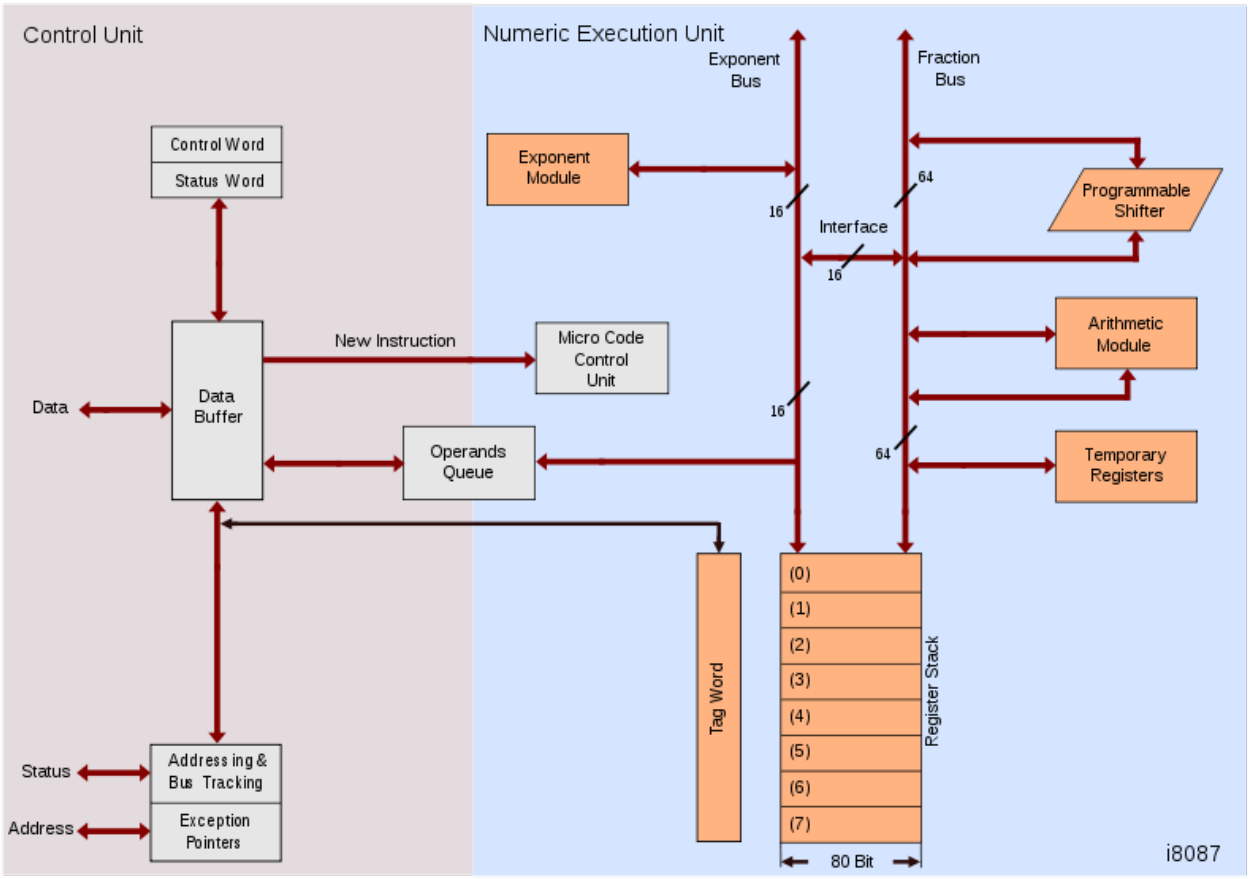
- Core 2 duo/i3/i5/i7 - 64bit processor
- OS – ubuntu 32bit/64bit OS
- Assembler used –nasm (the netwide assembler)
- Editor Used – gedit

THEORY:-

1. Introduction:

- 8087 was the first math coprocessor for 16-bit processors designed by Intel.
- It was built to pair with 8086 and 8088.
- The purpose of 8087 was to speed up the computations involving floating point calculations.
- Addition, subtraction, multiplication and division of simple numbers is not the coprocessor's job.
- It does all the calculations involving floating point numbers like scientific calculations and algebraic functions.
- By having a coprocessor, which performs all the calculations, it can free up a lot of CPU's time.
- This would allow the CPU to focus all of its resources on the other functions it has to perform.
- This increases the overall speed and performance of the entire system.
- This coprocessor introduced about 60 new instructions available to the programmer.
- All the mnemonics begin with "F" to differentiate them from the standard 8086 instructions.
- For e.g.: in contrast to ADD/MUL, 8087 provide FADD/FMUL.
- Math coprocessor is also called as:
  - Numeric Processor Extension (NPX)
  - Numeric Data Processor (NDP)
  - Floating Point Unit (FPU)

2. ARCHITECTURE OF 8087



The internal structure of 8087 coprocessor is divided into two major sections:

- **Control Unit (CU)**
- **Numerical Execution Unit (NEU)**

CONTROL UNIT (CU)

- It interfaces coprocessor to the microprocessor system bus.
- It also synchronize the operation of the coprocessor and the microprocessor.
- This unit has a Control Word, Status Word and Data Buffer.
- If an instruction is ESC instruction, then coprocessor executes it.
- If not, then microprocessor executes.

NUMERIC EXECUTION UNIT (NEU)

- This unit is responsible for executing all coprocessor instructions.
- It has an 8 register stack that holds the operands for instructions and result of instructions.
- The stack contains 8 registers that are 80-bits wide.
- Numeric data is transferred inside the coprocessor in two parts:
  - 64-bit mantissa bus
  - 16-bit exponent bus

3. INSTRUCTION SET

The 8087 instruction mnemonics begins with the letter F which stands for Floating point and distinguishes from 8086. These are grouped into Four functional groups. The 8087 detects an error condition usually called an exception when it executing an instruction it will set the bit in its Status register.

I. Data Transfer Instructions.
II. Arithmetic Instructions.
III. Compare Instructions.
IV. Transcendental Instructions.
(Trigonometric And Exponential)

I. DATA TRANSFERS INSTRUCTIONS

REAL TRANSFER

FLD	Load real
FST	Store real
FSTP	Store real and pop
FXCH	Exchange registers

INTEGER TRANSFER

FILD	Load integer
FIST	Store integer
FISTP	Store integer and pop

PACKED DECIMAL TRANSFER(BCD)

FBLD	Load BCD
FBSTP	Store BCD and pop

II. ARITHMETIC INSTRUCTIONS

Addition

FADD	Add real
FADDP	Add real and pop
FIADD	Add integer

Subtraction

FSUB	Subtract real
FSUBP	Subtract real and pop
FISUB	Subtract integer
FSUBR	Subtract real reversed
FSUBRP	Subtract real and pop
FISUBR	Subtract integer reversed

Multiplication

FMUL	Multiply real
FMULP	Multiply real and pop
FIMUL	Multiply integer

Advanced (Other Arithmetic Operations)

FABS	Absolute value
FCHS	Change sign
FPREM	Partial remainder
FPRNDINT	Round to integer
FSCALE	Scale
FSQRT	Square root
FXTRACT	Extract exponent and mantissa.

III. COMPARE INSTRUCTIONS

Comparison

FCOM	Compare real
FCOMP	Compare real and pop
FCOMPP	Compare real and pop twice
FICOM	Compare integer
FICOMP	Compare integer and pop
FTST	Test ST against +0.0
FXAM	Examine ST

IV. TRANSCENDENTAL INSTRUCTION (TRIGONOMETRIC AND EXPONENTIAL)

Transcendental

FPTAN	Partial tangent
FPATAN	Partial arctangent
F2XM1	2x - 1
FYL2X	Y log2X
FYL2XP1	Y log2(X+1)

Load Constant Instruction

FLDZ	Load +0.0
FLDI	Load+1.0
FLDPI	Load π
FLDL2T	Load log210
FLDL2E	Load log2e
FLDLG2	Load log102
FLDLN2	Load loge2

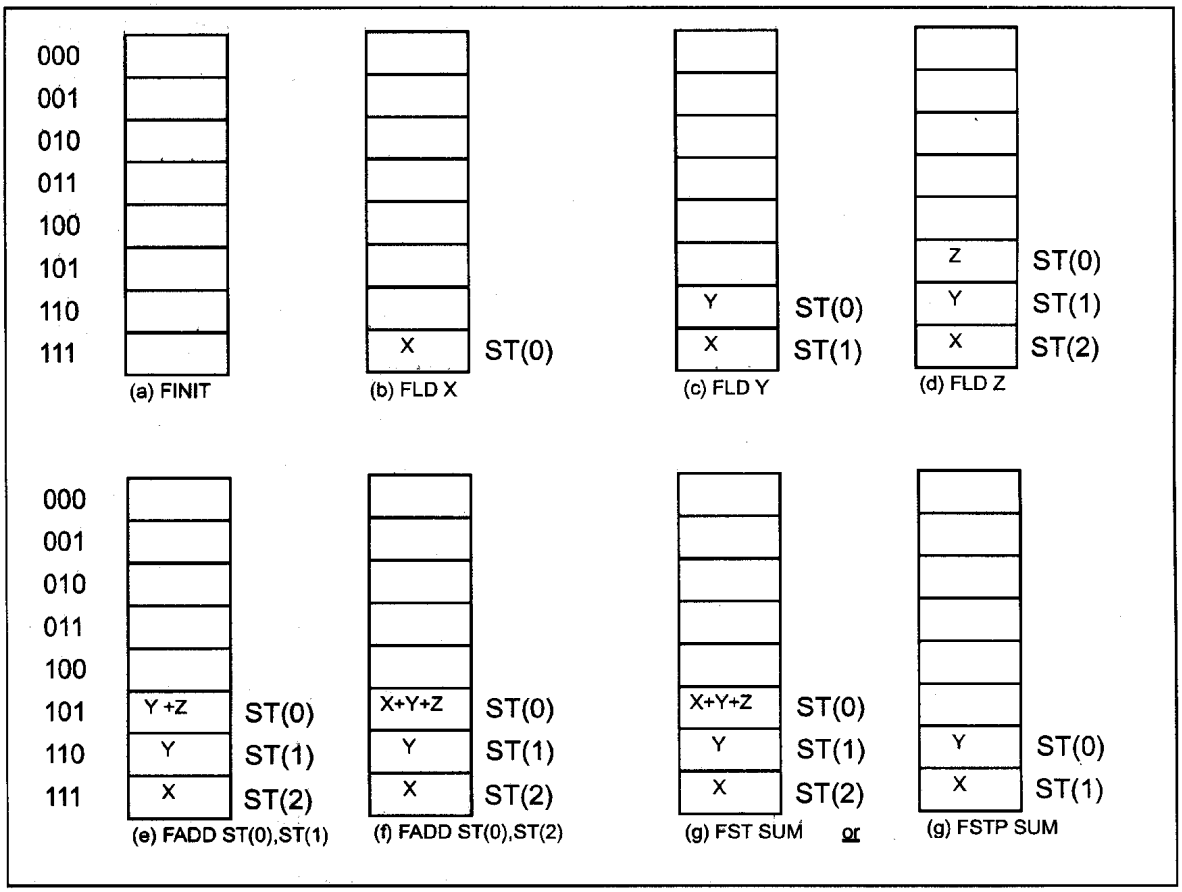
Write an 8087 program that loads three values for X, Y, and Z, adds them, and stores the result.

Solution:

```
finit           ;initialize the 8087 to start at the top of stack
fld     X       ;load X into ST(0). now ST(0)=X
fld     Y       ;load Y into ST(0). now ST(0)=Y and ST(1)=X
fld     Z       ;load Z into ST(0). now ST(0)=Z,ST(1)=Y,ST(2)=X
fadd    ST(1)   ;add Y to Z and save the result in ST(0)
fadd    ST(2)   ;add X to (Y+Z) and save it in ST(0)
fst     sum     ;store ST(0) in memory location called sum.
```

Now the same program can be written as follows:

```
finit
fld     X       ;load x, now ST(0) = x
fld     Y       ;load y, now ST(0)= y, ST(1) = x
fld     Z       ;load z, now ST(0)=z, ST(1)=y, ST(2)=x
fadd            ;adds y to z
fadd    ST(2)   ;adds x to (y + z)
fst     sum
```



CONCLUSION: -