1. Revisit 8086 microprocessor

The 8086 microprocessor contains 14 registers. Each register is 16 bits long. The 80386 through the Core2 microprocessors contain full 32-bit internal architectures. The Pentium 4 and Core2 also contain 64-bit registers.

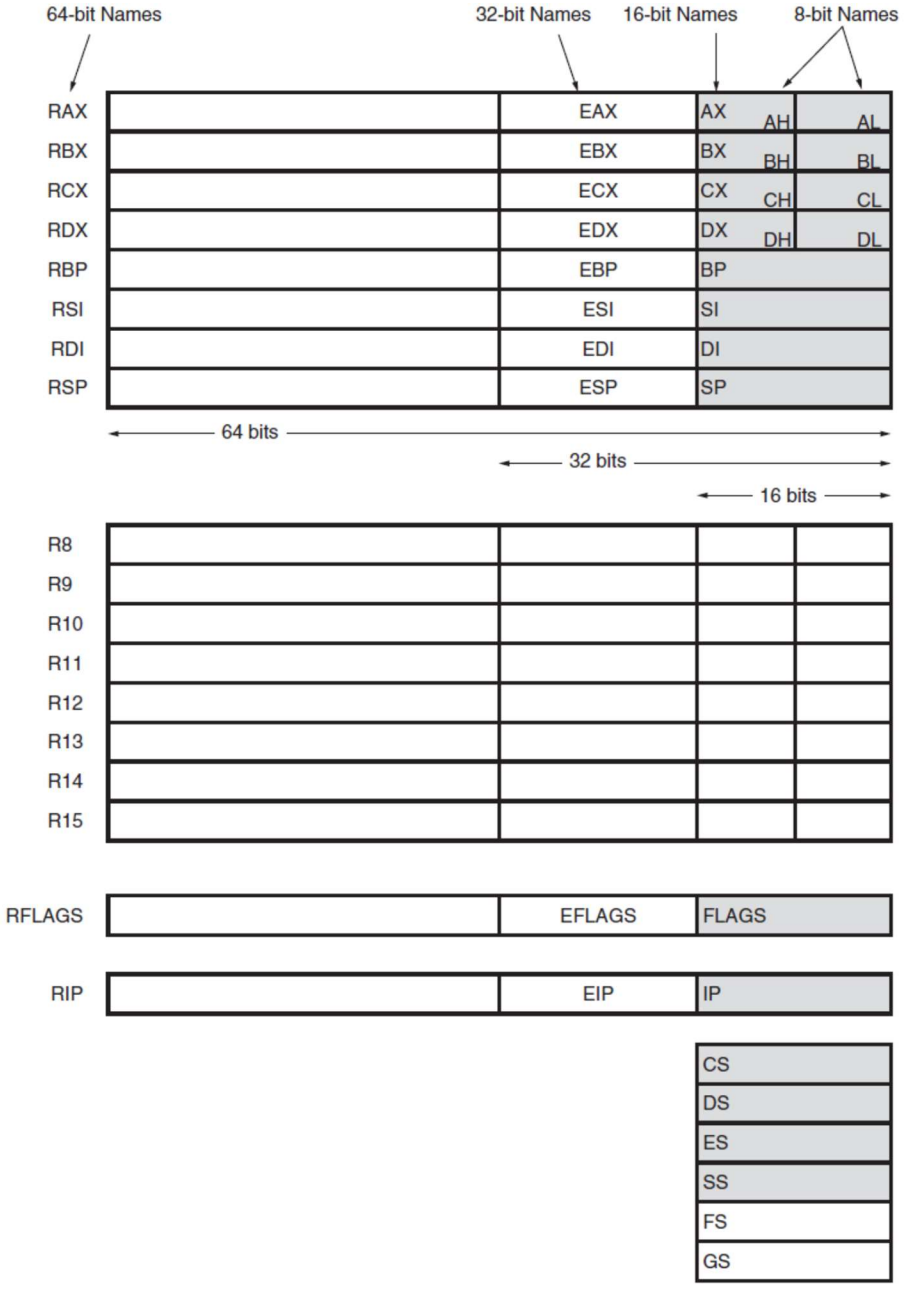

Fig 1: The programming model of the 8086 through the Core2 microprocessor including the 64-bit extensions.

I. General Purpose Registers

These registers hold various data sizes (bytes, words, or doublewords) and are used for almost any purpose, as dictated by a program.

The programming model of the 8086 through the Core2 microprocessor contains 8-, 16-, and 32-bit registers. The 8-bit registers are AH, AL, BH, BL, CH, CL, DH, and DL and are referred to when an instruction is formed using these two-letter designations.

For example,

ADD AL, AH

adds the 8-bit contents of AH to AL. (Only AL changes due to this instruction.) The

16-bit registers are AX, BX, CX, DX, contain a pair of 8-bit registers. An example is AX, which contains AH and AL. The 16-bit registers are referenced with the two-letter designations such as

AX.

For example,

ADD DX, CX

instruction adds the 16-bit contents of CX to DX. (Only DX changes due to this instruction.) The extended 32-bit registers are EAX, EBX, ECX, EDX.

For example,

ADD ECX, EBX

instruction adds the 32-bit contents of EBX to ECX. (Only ECX changes due to this instruction.) The 64-bit registers are designated as RAX, RBX, and so forth.

Table1: General purpose registers in the 8086 through the Core2 microprocessor including the 64-bit extensions.		
RAX	**Accumulator**	RAX is referenced as a 64-bit register (RAX), a 32-bit register (accumulator) (EAX), a 16-bit register (AX), or as either of two 8-bit registers (AH and AL). The accumulator is used for instructions such as multiplication, division, and some of the adjustment instructions. For these instructions, the accumulator has a special purpose, but is generally considered to be a multipurpose register.
RBX	**base index**	The BX register sometimes holds the offset address of a location in the memory system in all versions of the microprocessor.
RCX	**count**	holds the count for various instructions. also, can hold the offset address of memory data. Instructions that use a count are the repeated string instructions (REP/REPE/REPNE); and shift, rotate, and LOOP/LOOPD instructions. The shift and rotate instructions use CL as the count, the repeated string instructions use CX, and the LOOP/LOOPD instructions use either CX or ECX.
RDX	**data**	holds a part of the result from a multiplication or part of the dividend before a division.
RBP	**base pointer**	points to a memory location in all versions of the microprocessor for memory data transfers.
RDI	**destination index**	often addresses string destination data for the string instructions.
RSI	**source index**	The source index register often addresses source string data for the string instructions.
R8 through R15		These registers are only found in the Pentium 4 and Core2 if 64-bit extensions are enabled. Most applications will not use these registers until 64-bit processors are common.

II. Special Purpose Registers

The special-purpose registers include RIP, RSP, and RFLAGS; and the segment registers include CS, DS, ES, SS, FS, and GS.

Table1: General purpose registers in the 8086 through the Core2 microprocessor including the 64-bit extensions.

RIP	**instruction pointer**	The instruction pointer, which points to the next instruction in a program, is used by the microprocessor to find the next sequential instruction in a program located within the code segment. The instruction pointer can be modified with a jump or a call instruction.
RSP	**stack pointer**	The stack memory stores data through this pointer and is explained later in the text with the instructions that address stack data.
RFLAGS		indicate the condition of the microprocessor and control its operation.
CS	**code**	The code segment is a section of memory that holds the code (programs and procedures) used by the microprocessor. The code segment register defines the starting address of the section of memory holding code.
DS	**data**	The data segment is a section of memory that contains most data used by a program. Data are accessed in the data segment by an offset address or the contents of other registers that hold the offset address.
ES	**extra**	The extra segment is an additional data segment that is used by some of the string instructions to hold destination data.
SS	**stack**	The stack segment defines the area of memory used for the stack. The stack entry point is determined by the stack segment and stack pointer registers.
FS and GS		The FS and GS segments are supplemental segment registers available in the 8038 6–Core2 microprocessors to allow two additional memory segments for access by programs.

Example: Calulate : $(30+15)\times(575-225)+210$

Solution:

```
org     100h
mov     ax,30
add     15
mov     bx,575
sub     bx,225
mul     bx
add     210
ret
```

Exercise: complete the following Assembly code.

```
org 100h

.DATA

A    DW  11
B    DW  4
SUM DW  ?
DIFFERENCE  DW  ?
MULTIPLICATION DW ?
DIVISION    DW  ?
REMAINDER   DW  ?

.CODE

MAIN PROC

    MOV AX, @DATA
    MOV DS, AX

    ;add A and B and store the result into SUM


    ;subtract A and B and store the result into DIFFERENCE


    ;multiply A and B and store the result into MUTIPLICATION


    ;calculate A/B


 MAIN ENDP
END MAIN

ret
```