x86 Assembly Programming

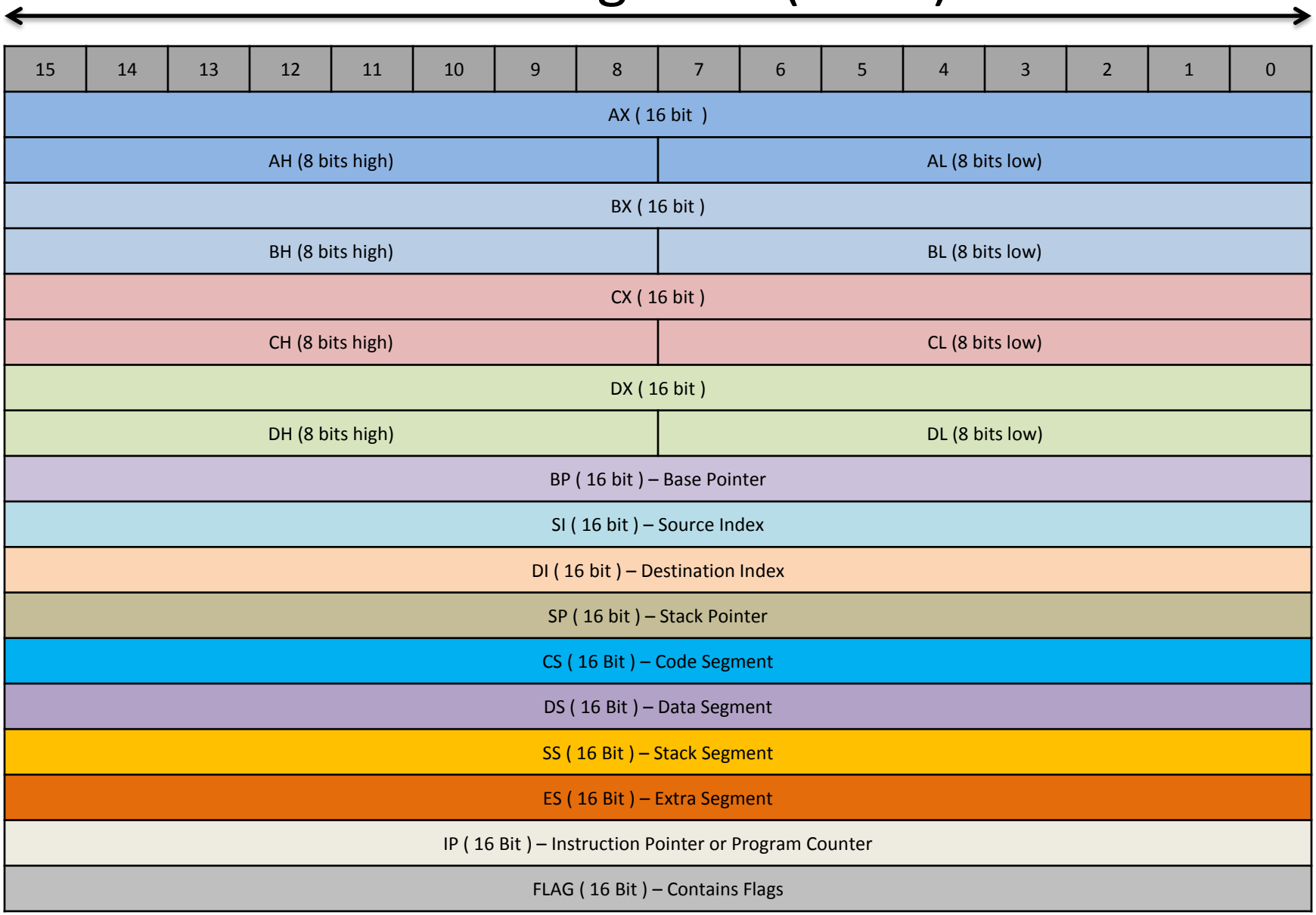
Lesson 3 – CPU

Ludvik Jerabek

16Bit CPU

- The Intel 80x86 compatible CPUs contain high speed storage areas called registers.
- Different 80x86 CPUs have a different numbers of registers.
- The 8086 microprocessor has a total of fourteen registers that are accessible to the programmer.
  - There are eight general purpose registers that can be used by the programmer for data manipulation.
  - All of the registers is 16 bits long and can contain a 16-bit binary number.
  - The AX, BX, CX and DX registers are general purpose registers often referred to as data registers.
  - The data registers can be treated as 16 bit registers or they can each be treated as two 8-bit registers using H and L in place of X.
  - For example AX → AH or AL , BX → BH or BL , CX → CH or CL , and DX → DH or DL where H are the high 8 bits and L are the low 8 bits.
  - The SP, BP, SI and DI registers are often known index/pointer registers.
  - The CS, DS, SS, ES registers are primarily for use with the 16 bit segmented memory model.
  - For more reading about the 8086 registers please see http://en.wikipedia.org/wiki/Intel_8086

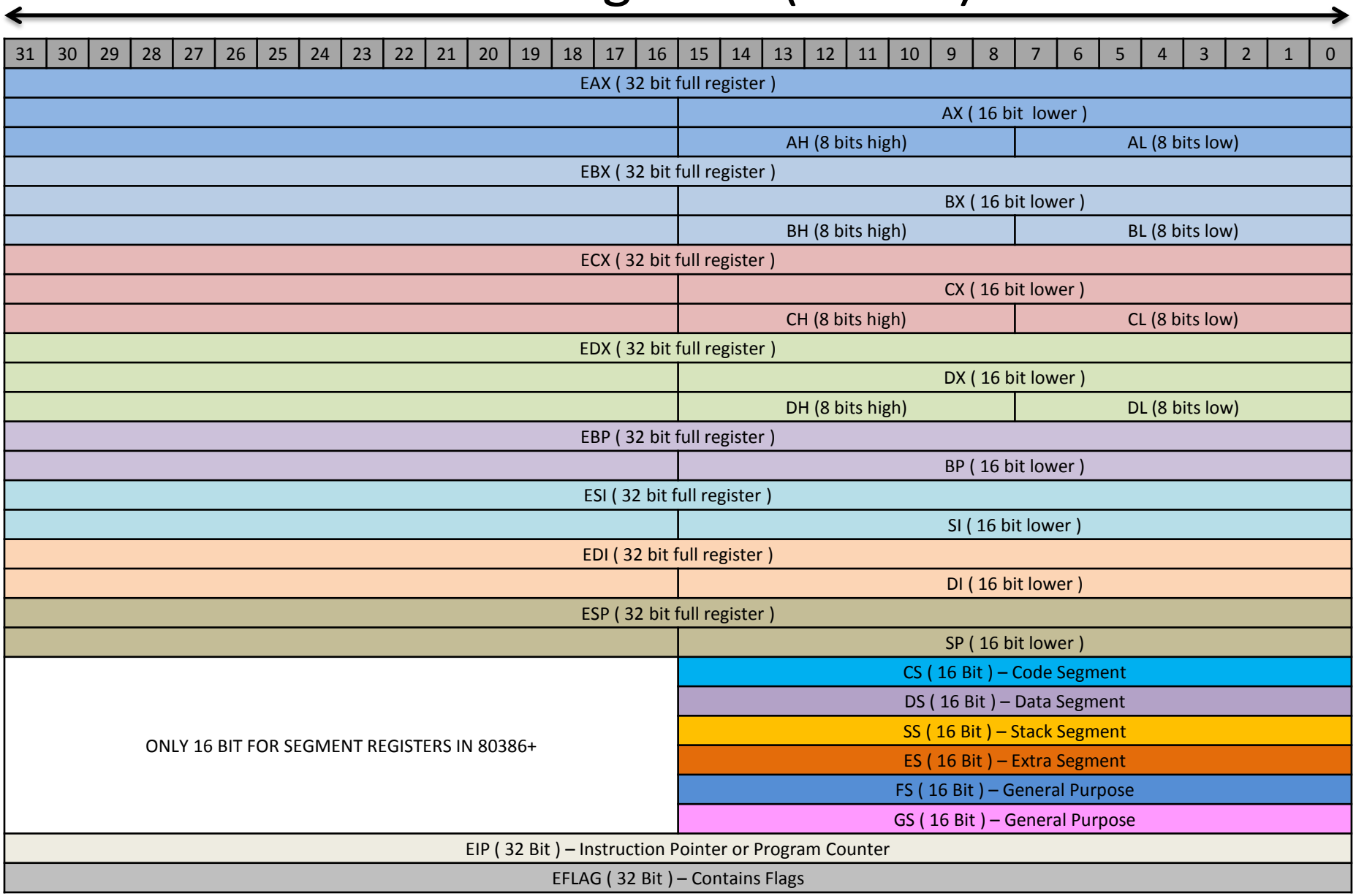
16 Bit Registers (8086)



32Bit CPU (Extended Registers)

- The Intel 80386 and higher CPUs have a numerous registers however below is a summary of the core set.
  - Just as with 16 bit general purpose registers like AX they have added a similar 32 bit registers named EAX, EBX, ECX, EDX
  - Also 32 bit versions of index/pointer registers have been added ESP, EBP, ESI, EDI
  - The data registers can be treated as 32 bit, 16 bit, 8 bit registers.
  - The 16 bit data registers are named AX, BX, CX, DX, SP, BP, SI, DI and the Low and High 8 bits can be accessed via AH, AL, BH, BL, CH, CL, DH, DL just as in 8086
  - There are also six segment registers CS, DS, SS, ES, FS, GS, on 80386 and higher primarily for use with the 16 bit segmented memory model.
  - For more reading about the 80386 registers please see http://en.wikipedia.org/wiki/Intel_80386

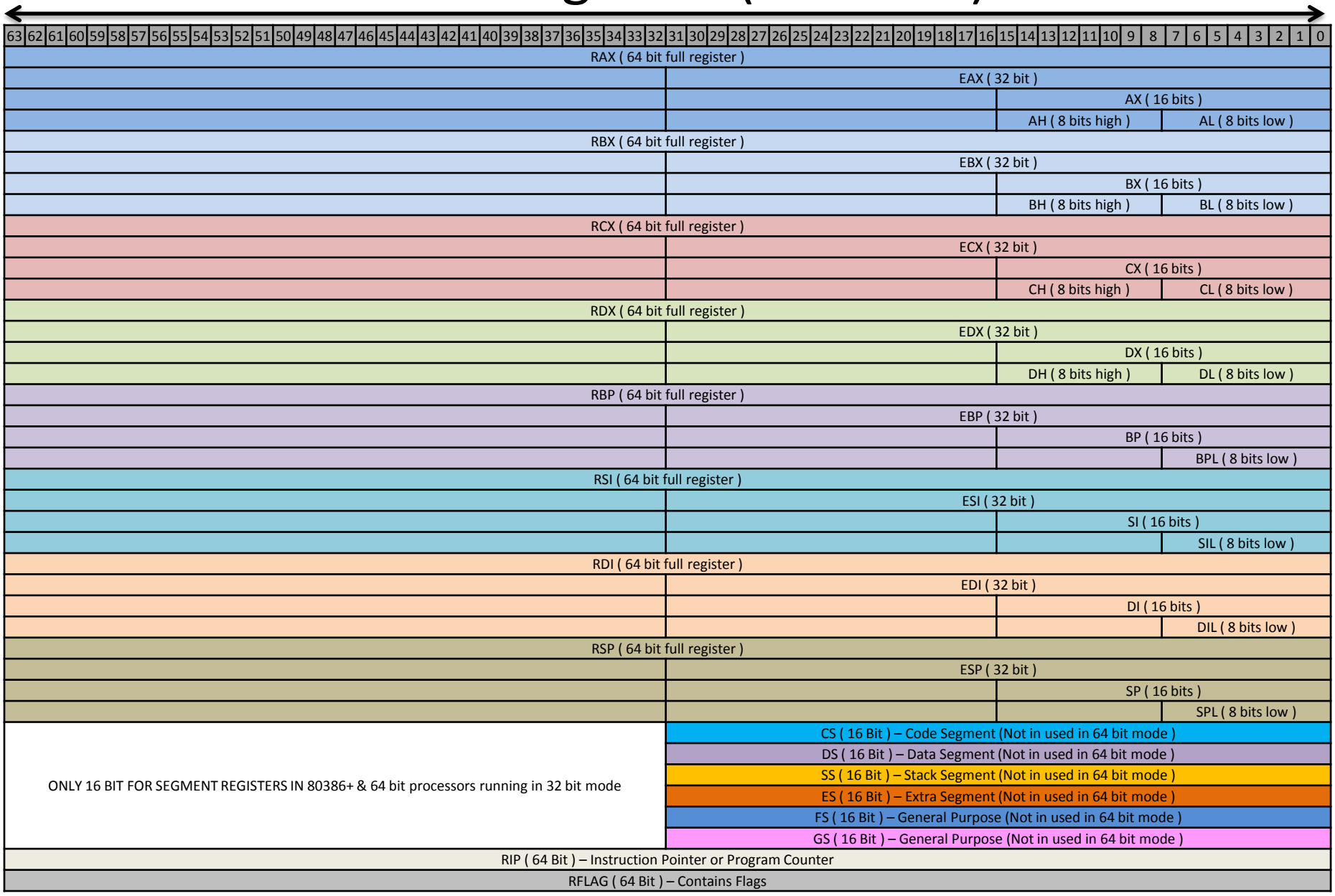
32 Bit Registers (80386)



64Bit CPU

- The Intel x64 CPU has a numerous registers below is a summary of the core set.
  - Just as with 16 bit and 32 bit general purpose registers AX and EAX they have added a similar 64 bit registers named RAX, RBX, RCX, RDX
  - Also 64 bit versions of index/pointer registers have been added RSP, RBP, RSI, RDI
  - The data registers can be treated as 64 bit, 32 bit, 16 bit, 8 bit registers.
  - The 32 bit data registers are named EAX, EBX, ECX, EDX, ESP, EBP, RSI, RDI just as in 80386
  - Likewise the 16 bit data registers are named AX, BX, CX, DX, BP, SP, SI, DI and Low and High 8 bits can be accessed via AH, AL, BH, BL, CH, CL, DH, DL
  - The segment registers CS, DS, SS, ES, FS, GS are only available when the processor is running in 32 bit mode
  - They have also added SIL, DIL, BPL, SPL which correspond to the lower 8 bits of SI, DI, BP, SP
  - For more reading about the x64 registers please see http://en.wikipedia.org/wiki/X86-64

64 Bit Registers (Intel x64)



Beating a dead horse ☺

- We've managed to take a look at the similarities of 16 bit, 32 bit, 64 bit CPUs from the Intel / AMD family of processors. It should be noted there are other specialized multimedia registers eg. MMX registers which are not covered in this tutorial.
- Registers, their names, and uses are probably the most difficult thing to commit to memory when first learning assembly however it's a necessary evil. With time and practice it will become second nature.
- Now that we have been introduced to the internals of various 8086+ CPUs its necessary to understand how to use these registers since not all registers are created equally.
- To cut down on the confusion of swapping between architectures 16, 32, and 64 bit. From this point forward we will strictly focus on 32 bit architecture since the majority of code today is running in 32 bit mode.
- Moving from 32 to 64 bit flat model should be fairly simple because of the similarity between the two architectures.

80386 Registers and Functions

Register	Bits	Purpose
EAX	32	General Use / Accumulator (Multiple, Divide, I/O)
EBX	32	General Use / Base (Pointer to base address)
ECX	32	General Use / Count (Count for loops and shifts)
EDX	32	General Use / Data (Multiple, Divide, I/O)
ESI	32	Source Index, Used to copy arrays of bytes or as an array index
EDI	32	Destination Index, destination address for array copy or as an array index
ESP	32	Stack Pointer, holds the address of the top of the stack
EBP	32	Base Pointer, used to store an address or position on the stack
CS	16	Stores the selector for the code segment (segmented memory model)
DS	16	Stores the selector for the data segment (segmented memory model)
ES	16	Stores the selector for the extra segment (segmented memory model)
SS	16	Stores the selector for the stack segment (segmented memory model)
FS	16	Stores the selector for a general segment (segmented memory model)
GS	16	Stores the selector for a general segment (segmented memory model)
EIP	32	Instruction Pointer, Stores the address of the next instruction to be executed
EFLAGS	32	Set of bit flags (such as overflow or carry) which provide information about an executed operation

Compiling, Linking, or Interpreting

Before introducing the 80x86 assembly syntax it's important to understand the difference between compiling, linking, and interpreting.

- Compilers
  - Compilers take source code "text" producing "object code" that are binary instructions the CPU can execute.
  - Assemblers are similar to compilers, except they convert "assembly text" to machine code. Since assembly code is almost one-to-one with machine code the job of the assembler is simpler than that of a compiler.
- Linkers
  - Linkers combine multiple pieces of "object code" created by a compiler/assembler and produces an executable or library. Executables on Windows/DOS based systems have the ".exe" extension and libraries have the ".lib" or ".dll" extension.
- Interpreters
  - Interperters are large programs that directly execute a text based source code or scripts. They usually run slower than compiled programs, however; can be useful for rapid prototyping where on-the-fly quick changes are needed. Perl, Bash, Basic, Lisp, Windows Script Host (WSH), and DOS batch/cmd are all examples of interpreted languages.

Assembly Syntax

- There are two main 80x86 assembly syntaxes*
  - Intel Syntax (the syntax we will use)
    - [instruction] [destination], [source]
      - To move a value of 255 into the eax register
      - mov eax, FF
    - Memory addressing base register uses []
      - mov eax, [ecx]
  - AT&T Syntax
    - [instruction] [source], [destination]
      - To move a value of 255 into the eax register
      - mov $0xFF, %eax
    - Memory addressing base register uses ()
      - mov (%ecx), %eax

*There are other differences, however they are minor and pretty easy to pick up if you know Intel or AT&T syntax

Wetting Your Assembly Appetite

- EAX or AX is commonly known as the accumulator register since it's often used to store the results of arithmetic operations.
  - add eax, 1123 (Changes full 32 bits EAX register)
  - add ax, 128 (Changes the 16 bits AX register)
  - add ah, 8 (Changes the high 8 bits of AX named AH)
  - add al, 16 (Changes the low 8 bits of AX named AL)
- All the data registers can be used for general use
  - mov ebx, 1 (Store 1 in ebx)
  - mov ecx, 2 (Store 2 in ecx)
  - mov edx, 3 (Store 3 in edx)
- Changes made to 16 and 8 bit high and low registers do not impact bit regions outside of the requested register
  - mov ebx, 0FFFFFFFFh
  - mov bh, 01h
  - mov bl, 01h
  - The value of BX is 0x0101 or decimal 257
  - The value of EBX is 0xFFFF0101

Try it!

Try your best to do it manually

Solve	Answer
Draw the relationships between RAX, EAX, AX, AH, AL.	
What is the size of the EBX register?	
What is the size of CS ?	
Is it possible to run in segmented mode on 64 bit systems in 64 bit mode?	
If EBX is 0x11223344 what is the value of BH and BX?	
How do you store the decimal value 32 into ECX ?	
After storing decimal 32 in to ECX how could you add 1 to the value?	