EE 5201 - Computer Architecture

Assembly Programming

H. Asela
Department of Electrical and Information Engineering
University of Ruhuna

Registers

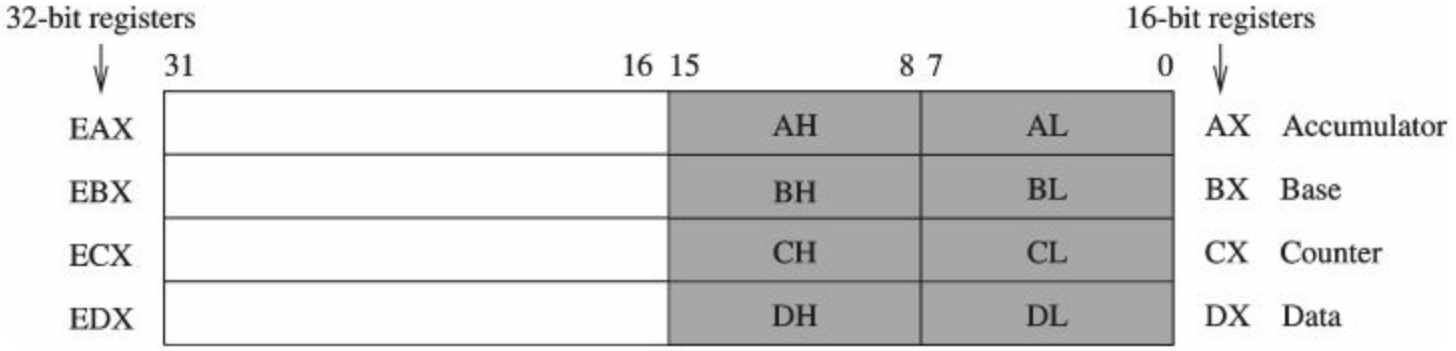
Super fast, Perform operations, Located on the chip and each core has its own set.

To speed up the processor operations, the processor includes some internal memory storage locations, called registers.

The registers store data elements, instructions and addresses for processing without having to access the memory. A limited number of registers are built into the processor chip.

A processor may contain different types of registers ; Data registers, Point registers, Index registers, Control registers and Segment registers

General Purpose Registers



These registers are used for arithmetic, logical, and other operations.

AX is the primary accumulator. Most of arithmetical operations are done in this register.

BX is known as the base register. The common use is to do array operations and indexed addressing.

CX is known as the count register. This register is used for counter purposes. (Ex: store the loop count in iterative operations.)

DX is known as the data register. This is used for data reservation.

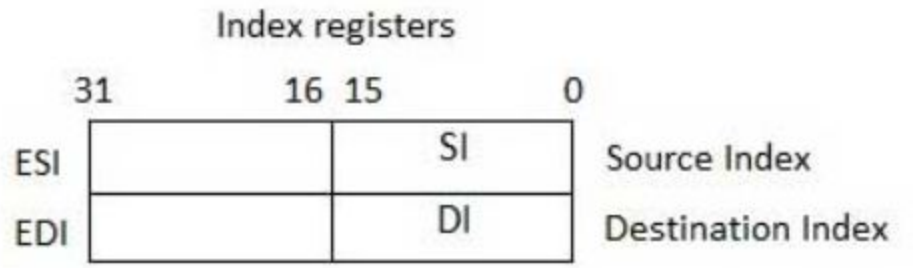
Point Registers

The pointer registers are 32-bit EIP, ESP, and EBP registers and corresponding 16-bit right portions IP, SP, and BP.

Usually BP is used for preserving space to use local variables. SP is used to point the current stack. IP denotes the current pointer of the running program.

Index Registers

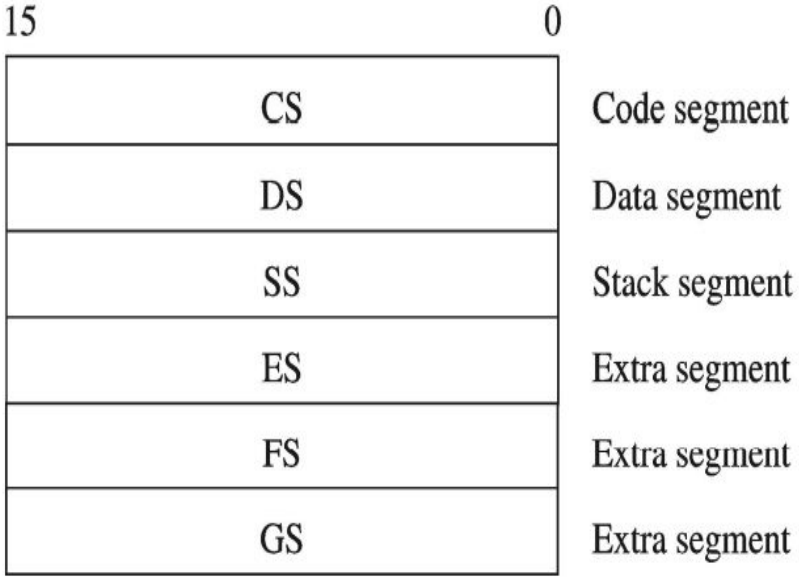
The 32-bit index registers, ESI and EDI, and their 16-bit rightmost portions. SI and DI, are used for indexed addressing and sometimes used in addition and subtraction.



Segment Registers

Segment registers are used to form addresses when you want to read/write to memory.

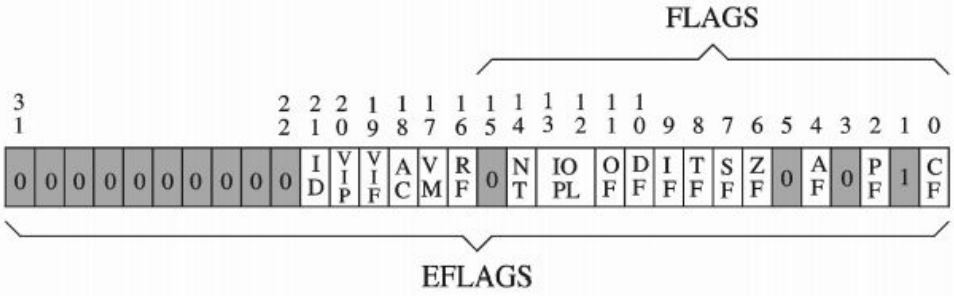
The x86 line of computers have 6 segment registers (CS, DS, ES, FS, GS, SS). They are totally independent of one another.



Flag Registers

Flag registers are used to store the current status of the processor.

It holds the value of which the programmers may need to access.



Status flags

CF = Carry flag
PF = Parity flag
AF = Auxiliary carry flag
ZF = Zero flag
SF = Sign flag
OF = Overflow flag

Control flags

DF = Direction flag

System flags

TF = Trap flag
IF = Interrupt flag
IOPL = I/O privilege level
NT = Nested task
RF = Resume flag
VM = Virtual 8086 mode
AC = Alignment check
VIF = Virtual interrupt flag
VIP = Virtual interrupt pending
ID = ID flag

Assembly Programming Syntax

Sections

An assembly program has three sections

- The data section - Used for declaring initialized data or constants. This data does not change at runtime.

  section .data

- The bss section - Used for declaring variables

  section .bss

- The text section - used for keep actual code. This section begins with declaration ***global _start***, which tells the kernel where the program execution begins.

  section .text

Comments

A comment is a piece of regular text that the assembler just discards when turning assembly code into machine code. However, comments improve the readability of the written code.

Comments are denoted by a semicolon **;**

Labels

A label is a symbol that represents the memory address of an instruction or data.It can be placed at the beginning of a statement.

During assembly, the label is assigned the current value of the active location counter and serves as an instruction operand.

Directives

Directives are instructions used by the assembler to help automate the assembly process and to improve program readability.

Mnemonics

mnemonic is a symbolic name for a single executable machine language instruction (an opcode).

There is at least one mnemonic defined for each machine language instruction.

Operands

Each assembly language statement is split into an opcode and an operand .

The opcode is the instruction that is executed by the processor. (Which is to be executed)

The operand is the data or memory location used to execute that instruction. (Parameters of the executed command)

Assembly Statements

Assembly language statements are entered one statement per line.

The format as follows

[label] mnemonic [operands] [;comment]

Ex: repeat: inc result ;increment result by 1

Allocating Storage for Initialized Data

DB (Define Byte) - Allocates 1 byte

DW (Define Word) - Allocates 2 bytes

DD (Define Doubleword) - Allocates 4 bytes

DQ (Define Quadword) - Allocates 8 bytes

DT (Define Ten Bytes) - Allocates 10 bytes

Allocating Storage for Uninitialized Data

RESB - Reserve a Byte

RESW - Reserve a Word

RESD - Reserve a Doubleword

RESQ - Reserve a Quadword

REST - Reserve Ten Bytes

System Calls

Programmatic way in which a computer program requests a service from the kernel. It is interface between the user space and the kernel space.

In Linux system call,

1. System call in EAX
2. Store the values, arguments and parameters in EBX, ECX, EDX etc.
3. Call the interrupt (0x80)
4. Any result will be stored in EAX

%eax	Name	%ebx	%ecx	%edx	%esx	%edi
1	sys_exit	int	-	-	-	-
2	sys_fork	struct pt_regs	-	-	-	-
3	sys_read	unsigned int	char *	size_t	-	-
4	sys_write	unsigned int	const char *	size_t	-	-
5	sys_open	const char *	int	int	-	-
6	sys_close	unsigned int	-	-	-	-

Interrupts

Interrupts are events to the processor signalizing that it has something to tell. It interrupt the processor.

Hardware interrupts occurs if one of the hardware inside the computer needs immediate processing. Delaying the process could cause unpredictable, or even, catastrophic effects. Keyboard interrupt is one of the example. If you press a key in the keyboard, you generate an interrupt. Keyboard chips notify the processor that they have a character to send.

Software interrupts occurs if the running program requests the program to be interrupted and do something else. It is usually like waiting the user input from keyboard, or may be request the graphic driver to initialize itself to graphic screen.

Thank you!