Computer Systems Organisation (CS2.201)

Summer 2021, IIIT Hyderabad

04 June, Friday (Lecture 6) – Instruction Set Architecture/Assembly Language Programming

Taught by Prof. Avinash Sharma

Instruction Format (contd.)

The instruction format is designed in such a way that there is always a unique decoding of a sequence of bytes into machine instruction.

Assembly Code Example

Consider the C code

```
int accum = 0;
int sum(int x, int y)
{
    int t = x + y;
    accum += t;
    return t;
}
```

Its assembler output is

```
sum:
    pushl %ebp
    movl  %esp, %ebp
    movl  12(%ebp), %eax
    addl  8(%ebp), %eax
    addl  %eax, accum
    popl  %ebp
    ret
```

The first two lines only update the status; push the current address on the stack and store the current function's address.
The next three lines carry out the addition of `x` and `y` and update the value of `accum`. At this point, `t` is stored in `%eax`, which is returned by default; therefore the `ret` instruction returns it.

Processor State

The PC (`%eip`) indicates the address of the next instruction to be executed and cannot be used for any other purpose.
The integer register file contains 8 locations storing 32-bit values, which can hold data, addresses, local variables or the return value.

The condition code registers hold information about the most recently executed arithmetic or logical operation. They are used to implement conditional branching.

Data Formats

"Word" in IA32 refers to a 16-bit (2-byte) datatype.

C declaration	Intel data type	Assembly code suffix	Size (bytes)
`char`	Byte	`b`	1
`short`	Word	`w`	2
`int`	Double word	`l`	4
`long int`	Double word	`l`	4
`long long int`	—	—	4
`char *`	Double word	`l`	4
`float`	Single precision	`s`	4
`double`	Double precision	`l`	8
`long double`	Extended precision	`t`	10/12

Figure 1: Sizes of C Datatypes in IA32

In x86-64, `long long int` is supported using the `quad` datatype, which needs 8 bytes.

Accessing Information: Registers

Out of the 8 registers, the first 6 are general-purpose (unless the instruction uses a specific register as a source or a destination).
The LS 2 bytes or 1 byte of the first four registers can be independently read or written by the byte operation instructions. For the remaining instructions, only the LS 2 bytes can be read or written.

In x86-64, there are 8 more registers, and all the registers have 64 bits.

Most instructions have one or more operands (source and destination).
There are three types of arguments: constant, register and memory. Values can be absolute, scaled or indexed.

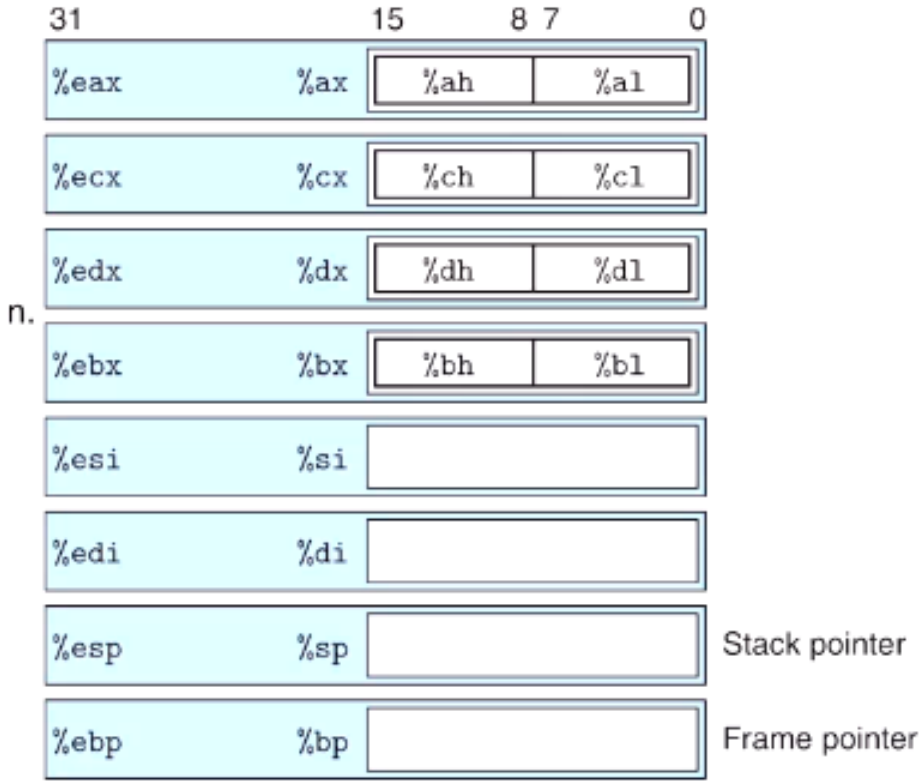

Figure 2: Registers in IA32

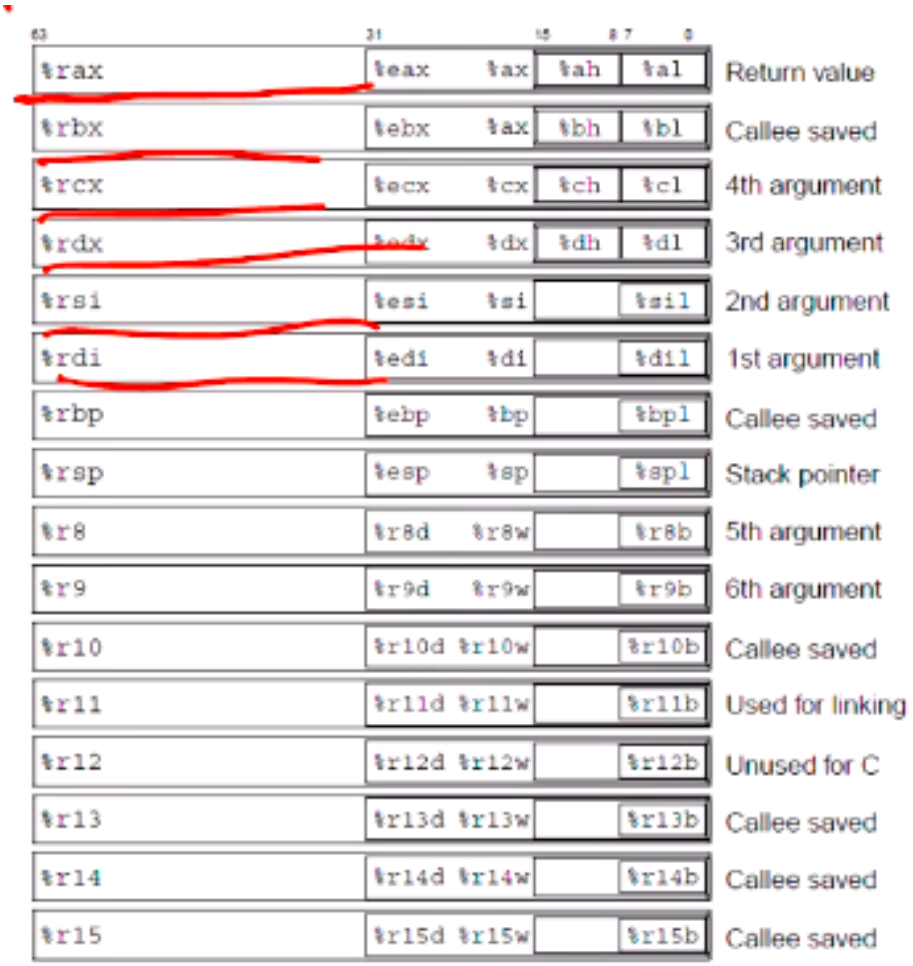

Figure 3: Registers in x86-64

Type	Form	Operand value	Name
Immediate	$\$Imm$	Imm	Immediate
Register	E_a	$R[E_a]$	Register
Memory	Imm	$M[Imm]$	Absolute
Memory	(E_a)	$M[R[E_a]]$	Indirect
Memory	$Imm(E_b)$	$M[Imm + R[E_b]]$	Base + displacement
Memory	(E_b, E_i)	$M[R[E_b] + R[E_i]]$	Indexed
Memory	$Imm(E_b, E_i)$	$M[Imm + R[E_b] + R[E_i]]$	Indexed
Memory	$(, E_i, s)$	$M[R[E_i] \cdot s]$	Scaled indexed
Memory	$Imm(, E_i, s)$	$M[Imm + R[E_i] \cdot s]$	Scaled indexed
Memory	(E_b, E_i, s)	$M[R[E_b] + R[E_i] \cdot s]$	Scaled indexed
Memory	$Imm(E_b, E_i, s)$	$M[Imm + R[E_b] + R[E_i] \cdot s]$	Scaled indexed

Figure 4: Operands