# x86 Assembly Language I

CMSC 313 Sections 01, 02

# 1.5 Historical Development

- Moore's Law (1965)
  - Gordon Moore, Intel founder
  - "The density of transistors in an integrated circuit will double every year."
- · Contemporary version:
  - "The density of silicon chips doubles every 18 months."

But this "law" cannot hold forever ...

2

## 1.5 Historical Development

- Rock's Law
  - Arthur Rock, Intel financier
  - "The cost of capital equipment to build semiconductors will double every four years."
  - In 1968, a new chip plant cost about \$12,000.

At the time, \$12,000 would buy a nice home in the suburbs.

An executive earning \$12,000 per year was "making a very comfortable living."

3

### Rock's Law

- In 2012, a chip plants under construction cost well over \$5 billion.

\$5 billion is more than the gross domestic product of some small countries, including Barbados, Mauritania, and Rwanda.

- For Moore's Law to hold, Rock's Law must fall, or vice versa. But no one can say which will give out first.

Processor Intro- duced Tech Intro		Max. Clock Frequency/ Technology at Introduction	Tran- sistors	Register Sizes <sup>1</sup>	Ext. Data Bus Size <sup>2</sup>	Max. Extern. Addr. Space	Caches	
9096	1976	8MHz	29 K	1600	16	1M9	None	
Intel 296	1982	12.5 MHz	134 K	16GP	16	10160	Note 2	
Intelligi DX Processor	1985	20160	275 K	22.60	32	468	Note 2	
inselellé DX Processor	1989	25 MHz	12M	32GP 80FPU	32	eca	Ltaka	
Persium Processor	1993	601840	31M	32 GP 80 FPU	64	468	L116 KS	
Pentium Pro Processor	n Pro Processor 1995 20		SSM	32GP 80FPU	64	61 (22	L2: 256 X or 512 XS	
Pertium II Processor	1997	200 1840	7M	SUGP SOFPU 64MBK	64	61 02	L1: 32 KS L2: 256 K or 512 KS	
Persium II Processor			82M	32 GP 80 FPU 64 MBK 129 XMM	ů4	64 02	L1: 32 KS L2: 512 KS	
Perdum II and Perdum II Xeon Processors	1999			32 GP 80 FPU 64 MMX 129 XMM	BOFPU GENNAL 128 XMM		L1: 32 KB L2: 256 KB	
Persium 4 Processor	2000	1.50 GHz, Irmi Nediusz Microarchitecture	eu	32 GP 80 FPU 64 MMX 129 XMM	64	64 02	Talk yasp Saecutos Trace Ca Lit 1958 I 256 KB	
Intel Xeon Processor	2001	1 70 GHz, Irral Nedikura Microarchitecture	eu	32-GP 80-FPU 64-MBX 129-XMM	64	64 02	Tak yap Sascutos Trace Ca L1: 869 I 51098	
Itel Xeon Processor	No.		Su.	22-GP 64 80 FPU 64 MBK 128 XMM		64 02	Tak yap Gascutos Tisce Ca Lt: 849 I S1299	
Persium MiProcessor	2009	1 80 GHz, Irmi Nediusz Microarchitecture	77 M	32 GP 80 FPU 64 MBK 129 XMM	ů4	+60	L1: 6902 L2:1 MB	
Park Perdum 4 Processor Supporting Hyper- Threading Technology at 90 nm process	2004	2.43 GHz. Irral historian Microarchinecture, HigherThreading Technology	135 M	30'GP BOFPU GENERAL 128 XMM	64	64 000	12K yap Sascutor Trace Ca L1: 16KB MB	

NOTE:

1. The register size and external data bus size are given in bits. Note also that each 32-bit general-purpose (GP) registers can be addressed as an 8-or a 16-bit data registers in all of the processors.

2. Internal data paths are 2 to 4 times wider than the external data bus for each processor.

Table 2-1. Key Features of Most Recent IA-32 Processors

Intel Processor	Date Intro- duced	Micro- architecture	Top-Bin Clock Fre- quency at Intro- duction	Tran- sistors	Register Sizes <sup>1</sup>	Syste m Bus Band- width	Max. Extern. Addr. Space	On-Die Caches <sup>2</sup>
Irasi Pentium M Processor 756 <sup>2</sup>	2004	Ital Penium M Processor	2.00 GHz	14018	GP: 32 FPU: 80 MMX: 64 XM8: 128	32 GB/s	4 GB	L1:64 KBL2:2 MB
Intel Core Duo Processo r T2600 <sup>2</sup>	2006	Improved Intel Pendium MProcessor Microarchitecture; Dual Cone; Istel Smart Cache, Advanced Thermal Manager	216 GHz	15268	GP: 32 FPU: 80 MMX: 64 XM82 128	5.3 GE/s	4 GB	L2 2MB (IMB Tasa)
Inali Asom ProcessorZlies series	2008	Intel Asset Microarchitecture; Intel Virtualization Technology	1.86 GHz - 800 MHz	eu	GP: 32 FPU: 80 MMC: 64 XMR: 128	Up to 4.2 GB/s	468	L1: 56932 <sup>4</sup> L2: 51263

	The	W- 2.2	. Key Feature	o of Most	Danant	Total 64 De		Control	
	Total		Micm-	Top-Bin	Tran-	Register	System	May.	On-Die
	Processor	Intro-	architec-ture	Fre-	sistor	Sizes	Bus/OP	Extern	Caches
		duced		quency	s		I Link	. Addr.	
				at Intro-			Speed	Space	
				duction					
	Intel Core/7-	2010	Intel Turbo Boost	266 GHz	383 M	GP. 32, 64		64 CB	L1:64 KB
	620M Processor		Technology, Istel microarchitecture			FPU: 80 MMX: 64			L2: 256KB L3: 4MB
	Processor					3885 128			DC 4000
			Westnerk; Dualcork:						
			HyperThreading						
			Technology; Ireal 64 Architecture:						
			Intel Virtualization						
			Technology, Integrated graphics						
	Intel Neon-	2010	Intel Turbo Boost	3.33 GHz	1.1R	GP:32,64	QPt 6.4	179	L1:64KB
	Processor \$460		Technology, Irrail			FPU:80	GTIs; 32	I	L2: 256KB
			microarchitecture			M00 64	GBN		L3: 1259
			coderame Westners:			38Mt 128			
			Six core:						
			HyperThreading Technology, Irrel 64						
			Intel Vinualization Technology						
	Intel Xeon-	2010		2.3% GHz	2.59		QPt 6.4	16 Tik	L1:64KB
	Processor7560		Technology, Intel			FPU:80	GTis;		L2: 256KB
			microarchitecture			M00: 64	Memory: 76		L3:26/B
			code name Nehalem;			38Mt 128	GB/s		
			Fight core; HiperThreading						
			Technology: Irrel 64						
			Architecture; Intel Virginianos						
			Technology.					l	
	Intel Core/7-	2011	Intel Turbo Boost	3.43 GHz	966M	GP: 32, 64	DMt 5 GT/s;	64 CER	L1:64 KB
	2600K		Technology, Irsel			FPU:80	Memory: 21	I	L2: 250KB
	Processor		microarchitecture code name Sandy			M00: 64 30M: 128	GBN	I	LX: SMB
			Bridge: Four core:			19Mt 256			
			HyperThreading Technology, Irrel 64					I	
			Architecture:					I	
			Intel Vinusization Technology,					I	
	Part Name	9044	Quickeync Video	250 GHz	_	GP 37 64	DM: SGTA:	. 100	11-64WD
		2011		ANUHE .					
_	ProcessorE3- 1290		Technology, Irsel microarchitecture			FPU:80 MMX:64	Memory: 21 GB/s	I	L3: 2569/ER L3: 5569
7	-		code name Sandy			28Mt 128		I	
			Bridge; Four core; HipperThreading			19Mt 256		I	
			Technology: kmil 64						

# This is a general depiction of a von Neumann system: These computers employ a fetch-decode-execute cycle to run programs as follows . . . Main Memory Main Me

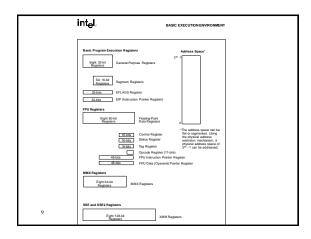


Figure 1.4 Advantas for agreement and credit disc.  1 Sec. Accordance for agreement and credit disc.  1 Sec. Accordance for agreement and credit disc.  2 Sec. Accordance for agreement and credit disc.  2 Sec. Accordance for agreement and credit disc.  3 Sec. Accordance for agreement and credit disc.  4 Sec. Accordance for agreement and credit disc.  4 Sec. Accordance for agreement and credit disc.  5 Sec. Accordance for agreement and credit disc.  1 Sec. Accordance for agreement and credit disc.  1 Sec. Accordance for agreement and credit disc.  1 Sec. Accordance for agreement and credit disc.  2 Sec. Accordance for agreement and credit disc.  2 Sec. Accordance for agreement and credit disc.  3 Sec. Accordance for agreement and credit disc.  4 Sec. Accordance for agreement and credit disc.  4 Sec. Accordance for agreement and credit disc.  5 Sec. Accordance for agreement and credit disc.  5 Sec. Accordance for agreement and credit disc.  6 Sec. Accordance for agreement and credit disc.  1 Sec. Accordance for agreement and credit disc.  2 Sec. Accordance for agreement and credit disc.  2 Sec. Accordance for agreement and credit disc.  3 Sec. Accordance for agreement and credit disc.  2 Sec. Accordance for agreement and credit disc.  3 Sec. Accordance for agreement and credit d
EXT—Incomplate for greated and reach date.   EXX—Incomplate for greated and reach greated and reach greated and reach greated great
EXT—Incomplate for greated and reach date.   EXX—Incomplate for greated and reach greated and reach greated and reach greated great
EXT-homodar for growth and reads does.   EXE-homodar does growth growth and reads does.   EXE-homodar does growth grow
Figure 34. Advanted General Purposes Rigigitar Names  Figure 35. Advanted General Purposes Rigigitar Names  10  EXX.—Accumulator for operatula and mosh done.  1 EXX.—Accumulator for operatula and mosh done.  2 EXX.—Accumulator for operatula and mosh done.  3 EXX.—Accumulator for operatula and mosh done.  4 EXX.—Accumulator for operatula and mosh done.  4 EXX.—Accumulator for operatula and mosh done.  5 EXX.—Accumulator for operatula and mosh done.  5 EXX.—Accumulator for operatula and mosh done.  6 EXX.—Accumulator for operatula and mosh done.  1 EXX.—Accumulator for operatula and mosh done.  1 EXX.—Accumulator for operatula and mosh done.  2 EXX.—Accumulator for operatula and mosh done.  3 EXX.—Accumulator for operatula and mosh done.  4 EXX.—Accumulator for operatula and mosh done.  5 EXX.—Accumulator for operatula and mosh done.  1 EXX.—Accumulator for operatula and mosh done.  1 EXX.—Accumulator for operatula and mosh done.  2 EXX.—Accumulator for operatula and mosh done.  3 EXX.—Accumulator for operatula and mosh done.  4 EXX.—Accumulator for operatula and mosh done.  5 EXX.—Accumulator for operatulator for operatural and mosh done.  6 EXX.—
Figure 3-4. Alexander Georgeat Proposes Register Materia  10  EAX—Accomment for operands and results data.  EEX—Accomment for the operands and results data.  EEX—Tourn to data in the Sta Segment.  EEX—Comment is data and the Operands.  EEX—Accomment is and and and topo operands.  EEX—Accomment is and and and operands.  EEX—Accomment is and and an operand not by the ISS register.  EEX—Accomment is and an operand on by the ISS register.  EEX—Comment is and the Comment on the Operand on by the ISS register.  EEX—Comment is and the Comment on the Operand on by the ISS register.  EEX—Accomment is and the Comment on the Operand on by the ISS register.  EEX—Accomment is and the Comment on the Operand on by the ISS register.  EEX—Accomment is and the Comment on the Operand on by the ISS register.
EAX.—Accountage for operands and treatly date.  EXX.—Accountage for operands and treatly date.  EXX.—Power red date in the CDS segment.  EXX.—Power red date in the CDS segment.  EXX.—Accountage of a designed sub-power.  EXX.—Accountage of
EAX—Accumulator for operands and results data.  EEX—Founce to data in the ES segment.  EXC—Counce for using ead loop operations.  EEX—Poince to data in the segment pointed to by the ES register, source pointer for string operations.  EEX—Poince to data in the segment pointed to by the ES register, source pointer for string operations to the segment pointed to by the ES register of the segment pointer for the s
EAX—Accumulator for operands and results data.  EEX—Founce to data in the ES segment.  EXC—Counce for using ead loop operations.  EEX—Poince to data in the segment pointed to by the ES register, source pointer for string operations.  EEX—Poince to data in the segment pointed to by the ES register, source pointer for string operations to the segment pointed to by the ES register of the segment pointer for the s
EAX—Accumulator for operands and results data.  EEX—Founce to data in the ES segment.  EXC—Counce for using ead loop operations.  EEX—Poince to data in the segment pointed to by the ES register, source pointer for string operations.  EEX—Poince to data in the segment pointed to by the ES register, source pointer for string operations to the segment pointed to by the ES register of the segment pointer for the s
EAX—Accumulator for operands and results data.  EEX—Founce to data in the ES segment.  EXC—Counce for using ead loop operations.  EEX—Poince to data in the segment pointed to by the ES register, source pointer for string operations.  EEX—Poince to data in the segment pointed to by the ES register, source pointer for string operations to the segment pointed to by the ES register of the segment pointer for the s
EEC.—Counter for string and loop operations.  EEC.—Counter for string and loop operations.  EEC.—I O pointer.  ES.—Pointer to data in the segment pointed to by the DS register, source pointer for string operations.  EEC.—Pointer to data for destination in the segment pointed to by the ES register destination pointer for string operations.  EEC.—Source to data for destination in the segment pointed to by the ES register destination pointer for string operations.
EEC.—Counter for string and loop operations.  EEC.—Counter for string and loop operations.  EEC.—I O pointer.  ES.—Pointer to data in the segment pointed to by the DS register, source pointer for string operations.  EEC.—Pointer to data for destination in the segment pointed to by the ES register destination pointer for string operations.  EEC.—Source to data for destination in the segment pointed to by the ES register destination pointer for string operations.
EEC.—Counter for string and loop operations.  EEC.—Counter for string and loop operations.  EEC.—I O pointer.  ES.—Pointer to data in the segment pointed to by the DS register, source pointer for string operations.  EEC.—Pointer to data for destination in the segment pointed to by the ES register destination pointer for string operations.  EEC.—Source to data for destination in the segment pointed to by the ES register destination pointer for string operations.
EEC.—Counter for string and loop operations.  EEC.—Counter for string and loop operations.  EEC.—I O pointer.  ES.—Pointer to data to the segment pointed to by the DS register, source pointer for string operations.  EEC.—Pointer to data for declarations in the segment pointed to by the ES register, source pointer for string operations.  EEC.—Pointer to data for declarations in the segment pointed to by the ES register, declaration pointer for string operations.  ES.—Sack pointer (to the SS segment).
EEC.—Counter for string and loop operations.  EEC.—Counter for string and loop operations.  EEC.—I O pointer.  ES.—Pointer to data to the segment pointed to by the DS register, source pointer for string operations.  EEC.—Pointer to data for declarations in the segment pointed to by the ES register, source pointer for string operations.  EEC.—Pointer to data for declarations in the segment pointed to by the ES register, declaration pointer for string operations.  ES.—Sack pointer (to the SS segment).
EEC.—Counter for string and loop operations.  EEC.—Counter for string and loop operations.  EEC.—I O pointer.  ES.—Pointer to data to the segment pointed to by the DS register, source pointer for string operations.  EEC.—Pointer to data for declarations in the segment pointed to by the ES register, source pointer for string operations.  EEC.—Pointer to data for declarations in the segment pointed to by the ES register, declaration pointer for string operations.  ES.—Sack pointer (to the SS segment).
EEC.—Counter for string and loop operations.  EEC.—Counter for string and loop operations.  EEC.—I O pointer.  ES.—Pointer to data to the segment pointed to by the DS register, source pointer for string operations.  EEC.—Pointer to data for declarations in the segment pointed to by the ES register, source pointer for string operations.  EEC.—Pointer to data for declarations in the segment pointed to by the ES register, declaration pointer for string operations.  ES.—Sack pointer (to the SS segment).
EEC.—Counter for string and loop operations.  EEC.—Counter for string and loop operations.  EEC.—I O pointer.  ES.—Pointer to data in the segment pointed to by the DS register, source pointer for string operations.  EEC.—Pointer to data for destination in the segment pointed to by the ES register destination pointer for string operations.  EEC.—Source to data for destination in the segment pointed to by the ES register destination pointer for string operations.
EEC.—Counter for string and loop operations.  EEC.—Counter for string and loop operations.  EEC.—I O pointer.  ES.—Pointer to data to the segment pointed to by the DS register, source pointer for string operations.  EEC.—Pointer to data for declarations in the segment pointed to by the ES register, source pointer for string operations.  EEC.—Pointer to data for declarations in the segment pointed to by the ES register, declaration pointer for string operations.  ES.—Sack pointer (to the SS segment).
<ul> <li>ESL—Poster to data the segment pointed by the ES register; some pointer for string operations.</li> <li>EEL—Poster to the Confidencies in the segment pointed to by the ES register;</li> <li>EEL—Poster to the Confidencies in the segment pointed to by the ES register;</li> <li>ESS—Sack pointer (in the SS expected).</li> </ul>
<ul> <li>ESP—Stack pointer (in the SS segment).</li> </ul>
EBP—Pointer to data on the stack (in the SS segment).
"Hello World" in Linux Assembly
Hello World III Linux Assembly
Use your favorite UNIX editor (vi, emacs, pico,)
Assemble using NASM on
Programme regular introduction
gl.umbc.edu nasm -f elf hello.asm
gl.umbc.edu nasm -f elf hello.asm     NASM documentation is on-line.
gl.umbc.edu nasm -f elf hello.asm
gl.umbc.edu nasm -f elf hello.asm  • NASM documentation is on-line.  • Need to "load" the object
gl.umbc.edu nasm -f elf hello.asm  • NASM documentation is on-line.  • Need to "load" the object file ld hello.o -melf_i386

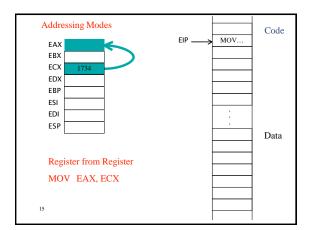
	x86 Addressing Modes	
13		

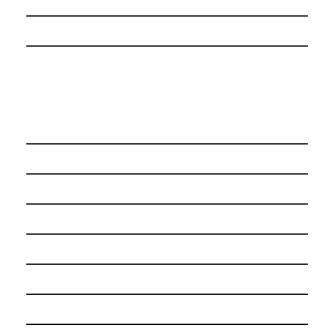
# 80x86 Addressing Modes

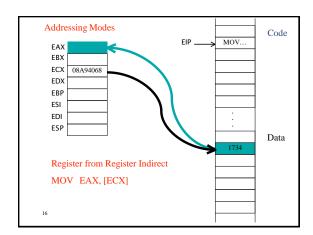
- We want to store the value 1734h.
- The value 1734h may be located in a register or in memory.
- The location in memory might be specified by the code, by a register, ...
- Assembly language syntax for MOV:

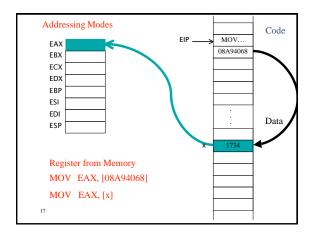
MOV DEST, SOURCE

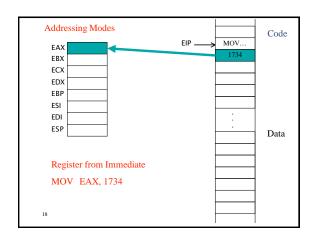
14

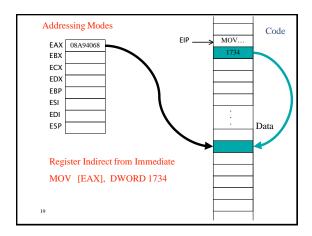


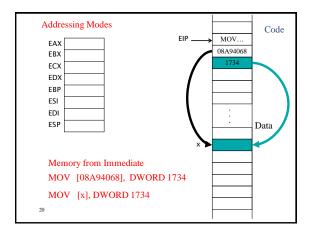












# **Notes on Addressing Modes**

• More complicated addressing modes later:

MOV EAX, [ESI+4\*ECX+12]

- Figures not drawn to scale. Constants 1734h and 08A94068h take 4 bytes (little endian).
- Some addressing modes are not supported by some operations.
- Labels represent addresses not contents of memory.

21