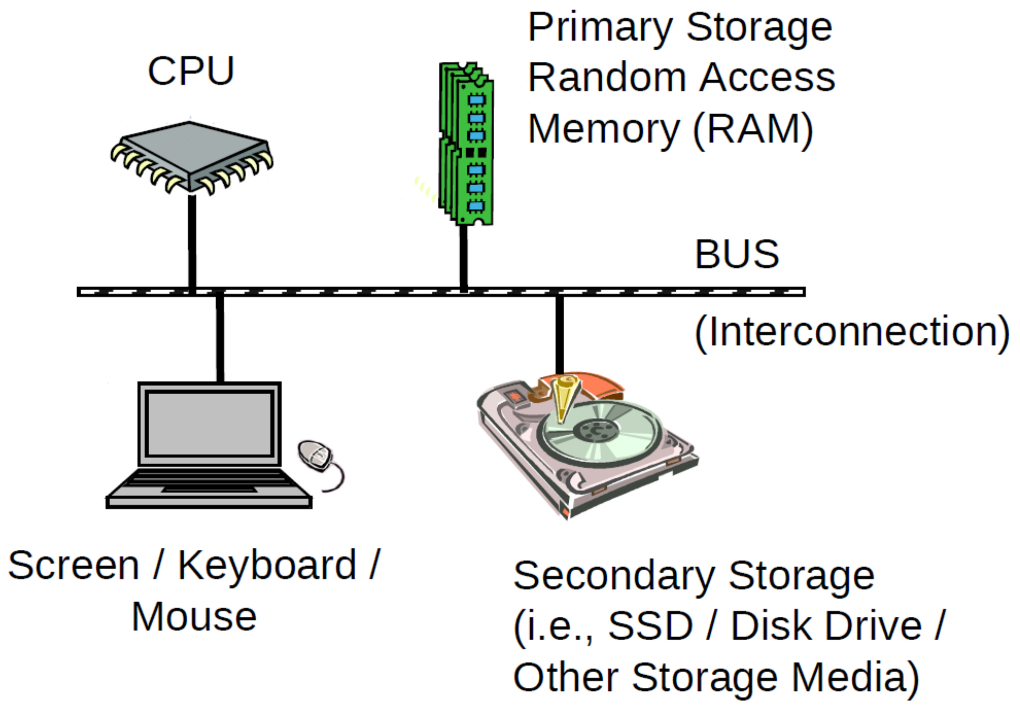
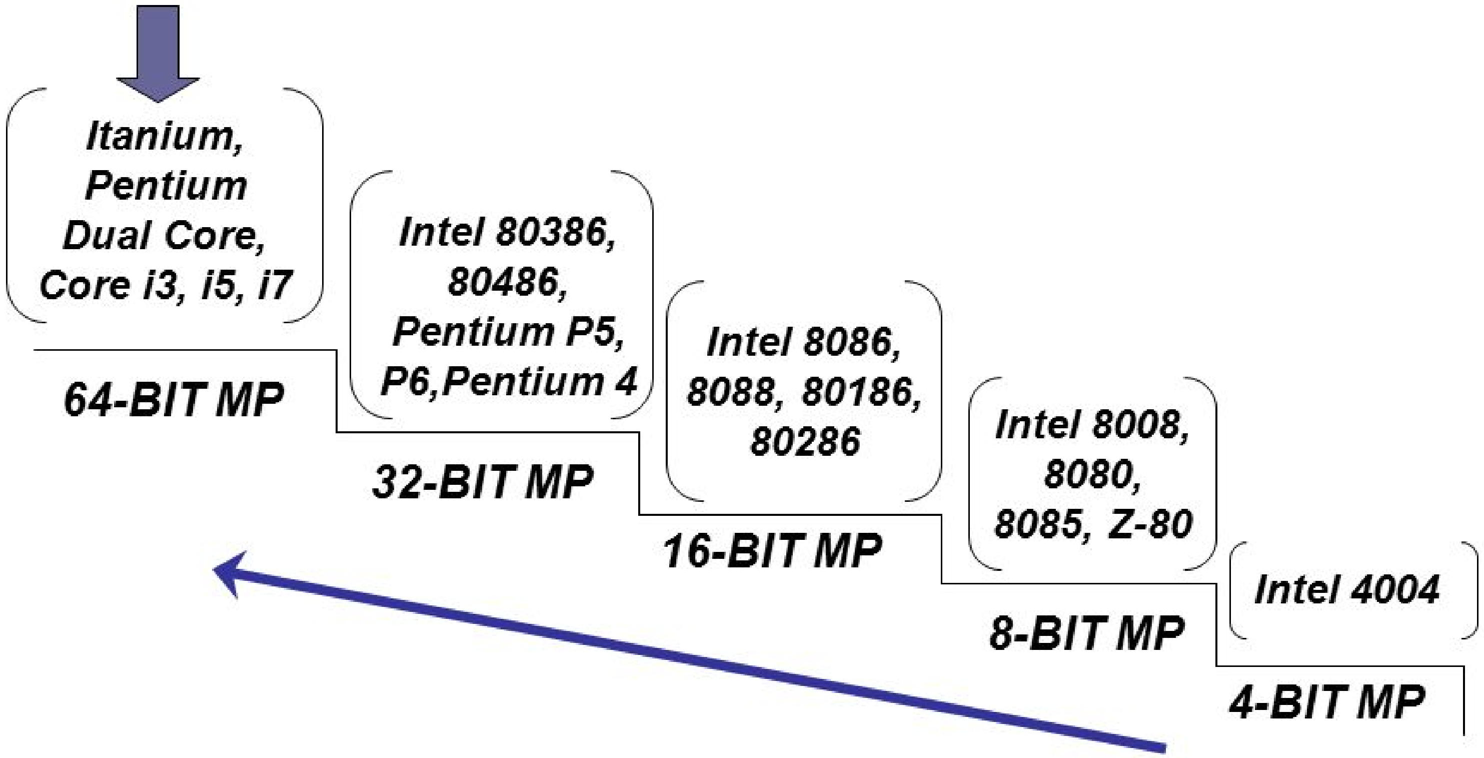
Architecture Overview

The basic components of a computer include

- Central Processing Unit (CPU),
- Primary Storage or Random Access Memory (RAM),
- Secondary Storage,
- Input/Output devices

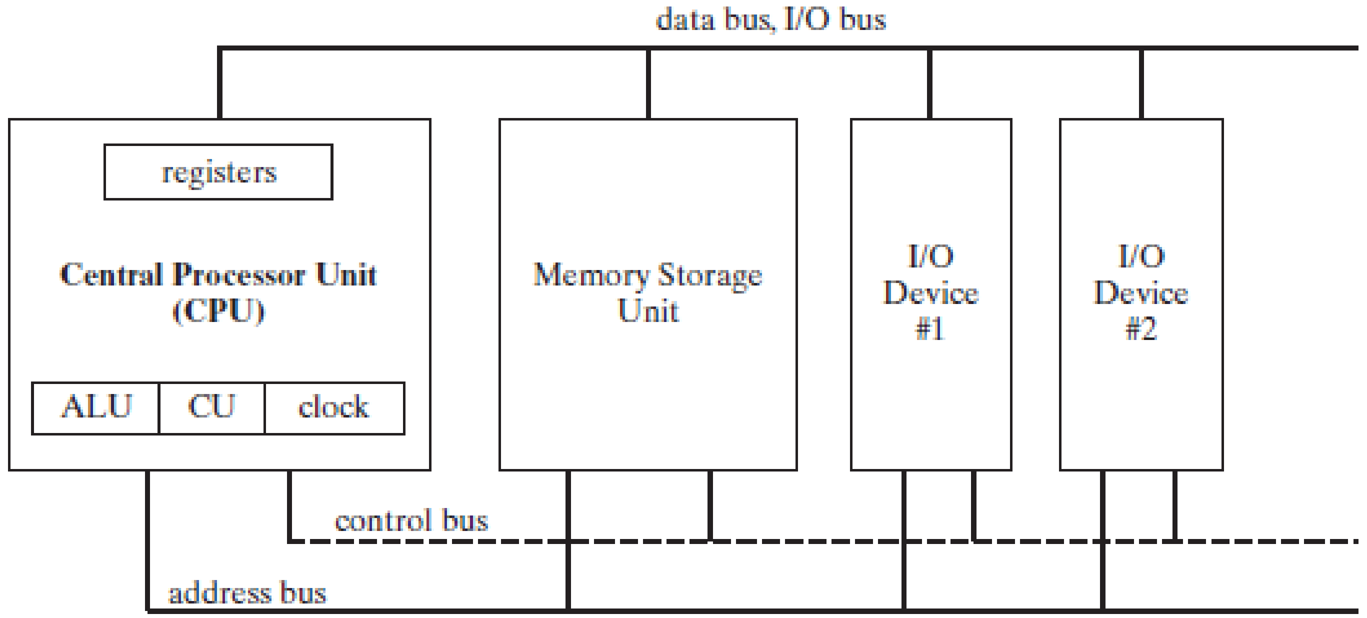


Architecture Overview


Itanium, Pentium Dual Core, Core i3, i5, i7
64-BIT MP
Intel 80386, 80486, Pentium P5, P6,Pentium 4
32-BIT MP
Intel 8086, 8088, 80186, 80286
16-BIT MP
Intel 8008, 8080, 8085, Z-80
8-BIT MP
Intel 4004
4-BIT MP

Architecture Overview

Inside CPU

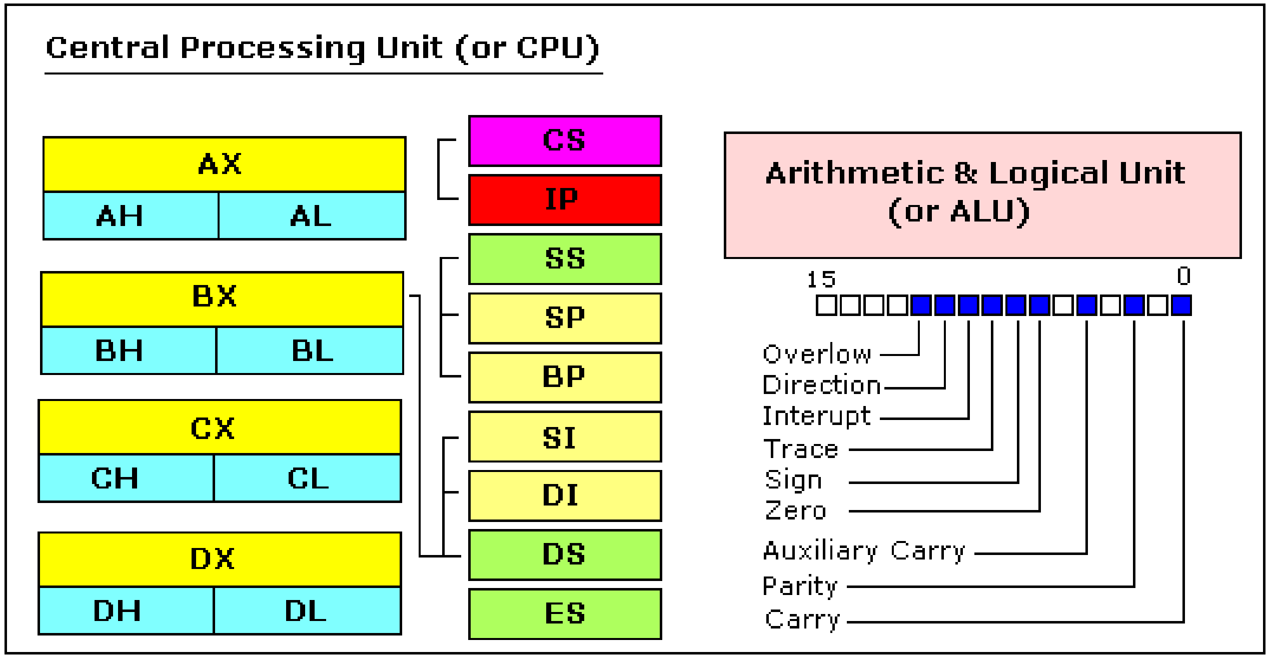


The CPU chip includes the Arithmetic Logic Unit (ALU) which is the part of the chip that actually performs the arithmetic and logical calculations.

A CPU register, or just register, is a temporary storage

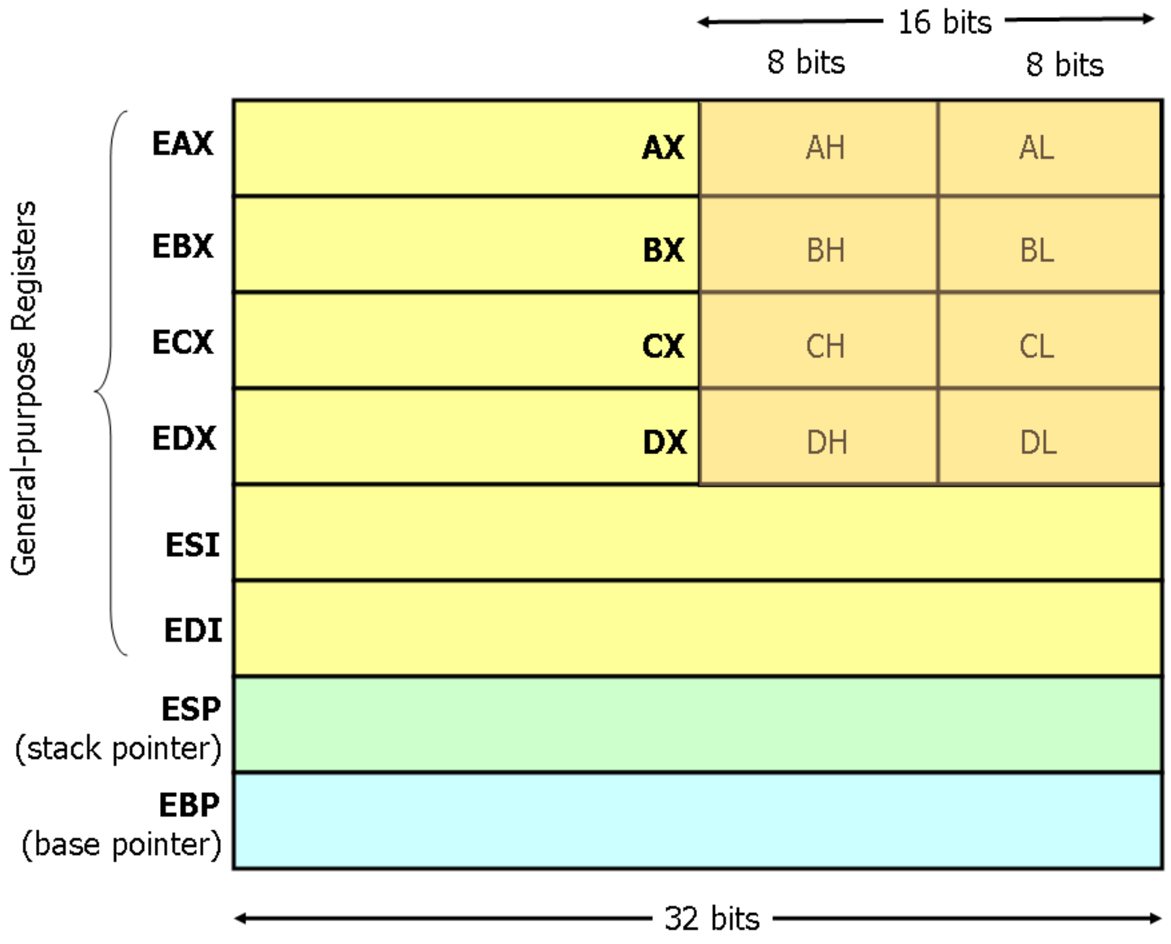
Architecture Overview

16-bit registers (8088/8086)



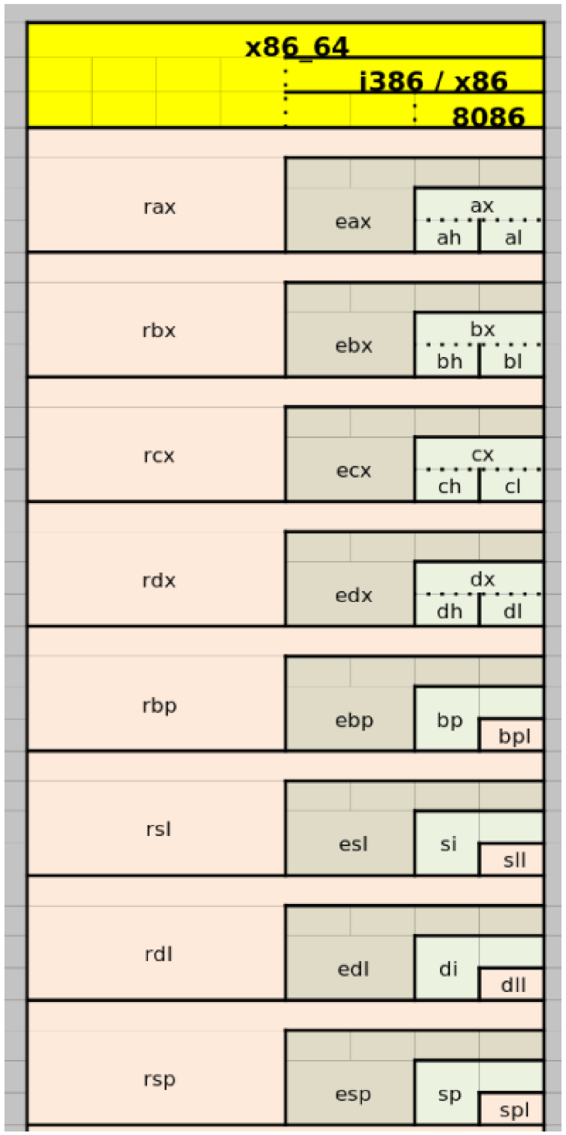
Architecture Overview

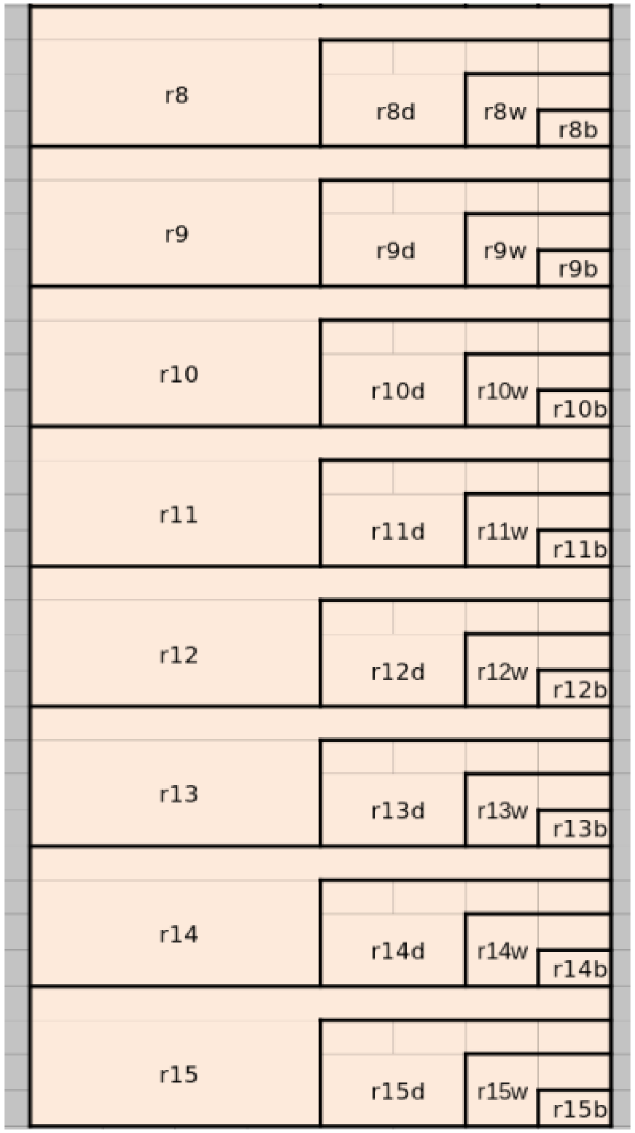
32-bit registers (80386, Pentium, Pentium II, Pentium III, P4)



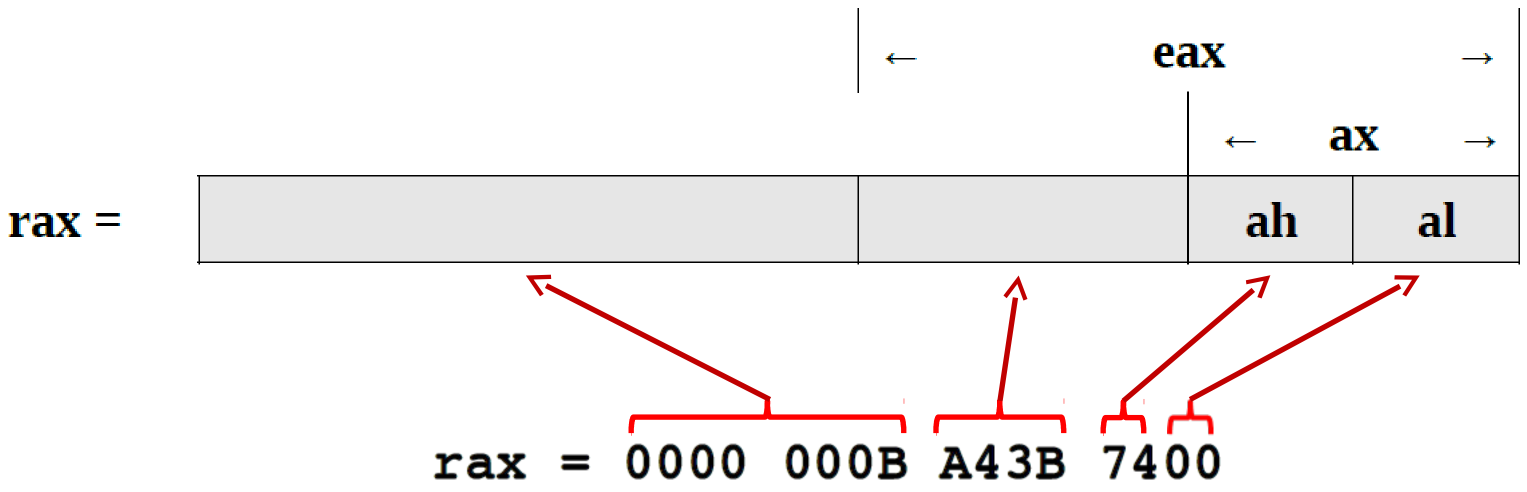
Architecture Overview

64-bit registers (P4, Core I) Partially displayed





Architecture Overview

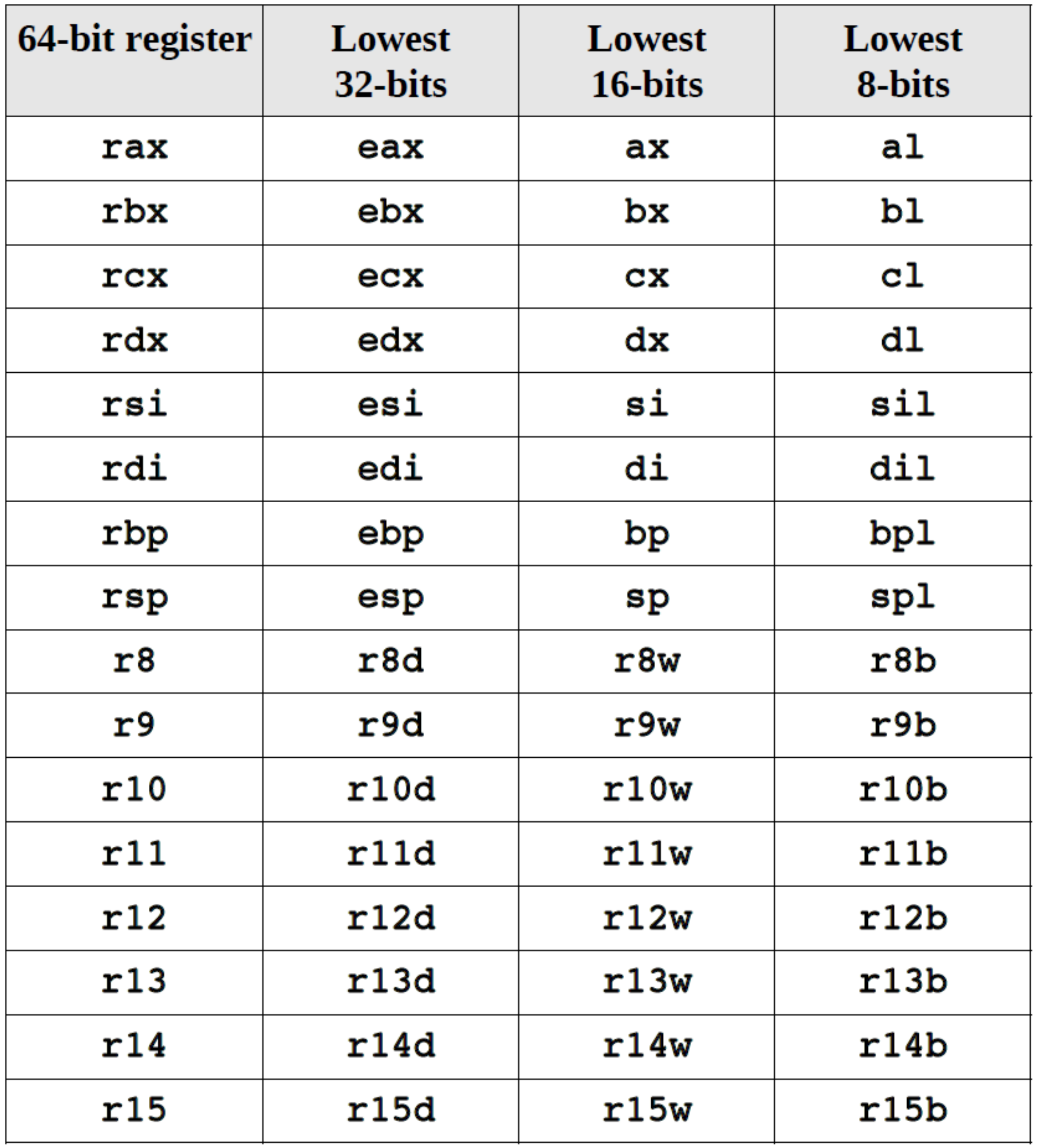


al = 00h
ah = 74h
ax = 0074h

eax = A43B7400h

Architecture Overview

General Purpose Registers (GPRs)



64-bit register	Lowest 32-bits	Lowest 16-bits	Lowest 8-bits
rax	eax	ax	al
rbx	ebx	bx	bl
rcx	ecx	cx	cl
rdx	edx	dx	dl
rsi	esi	si	sil
rdi	edi	di	dil
rbp	ebp	bp	bpl
rsp	esp	sp	spl
r8	r8d	r8w	r8b
r9	r9d	r9w	r9b
r10	r10d	r10w	r10b
r11	r11d	r11w	r11b
r12	r12d	r12w	r12b
r13	r13d	r13w	r13b
r14	r14d	r14w	r14b
r15	r15d	r15w	r15b

Architecture Overview

Special Function Registers

Stack Pointer Register (rsp) is used to point to the current top of the stack. The rsp register should not be used for data or other uses.

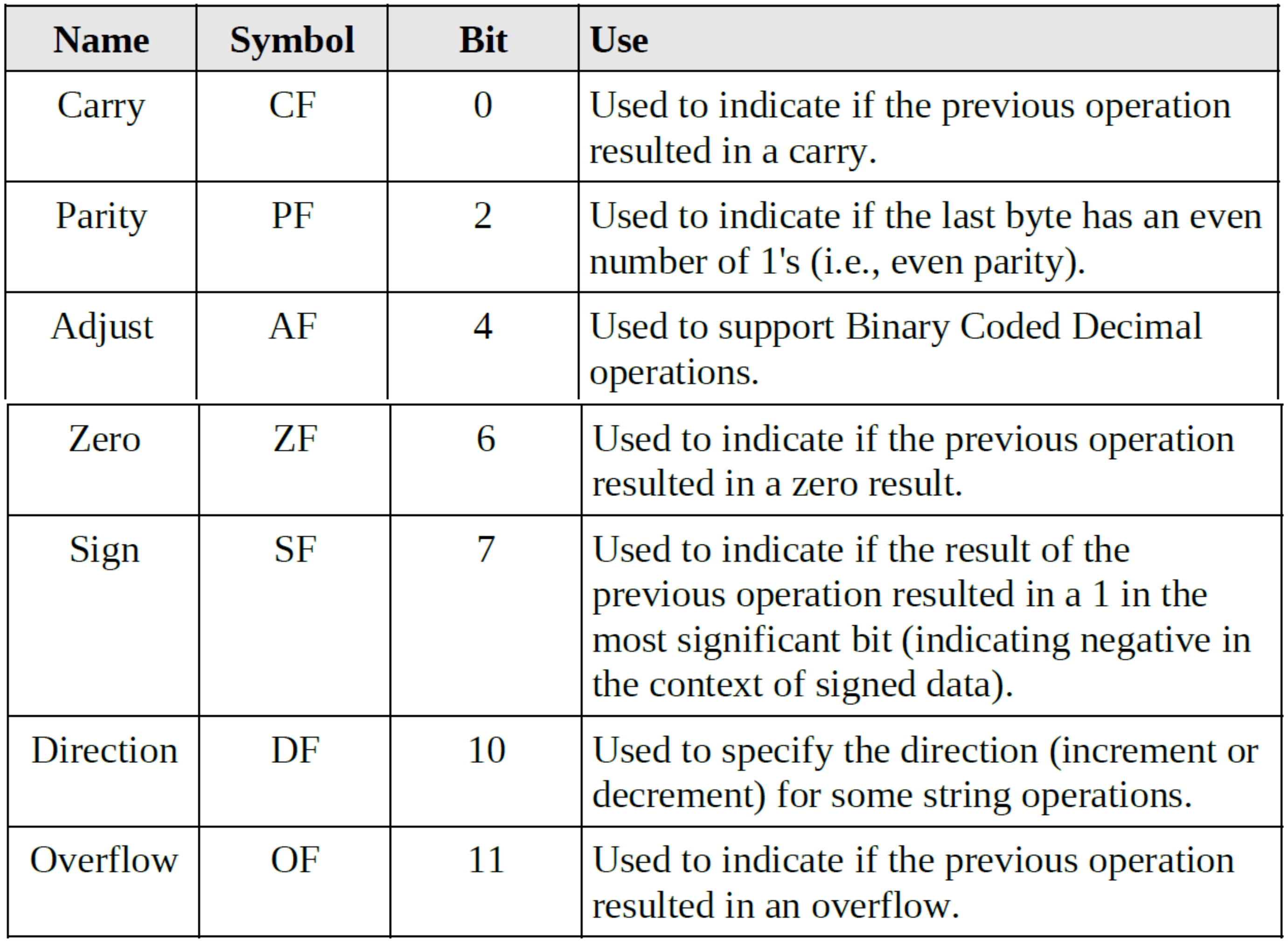
Base Register Pointer (rbp) is used as a base pointer during function calls

Instruction Pointer (rip) is used by the CPU to point to the next instruction to be executed. Specifically, since the rip points to the next instruction, that means the instruction being pointed to by rip, and shown in the debugger, has not yet been executed.

The flag register (rFlags) is used for status and CPU control information. The rFlag register is updated by the CPU after each instruction and not directly accessible by programs. This register stores status information about the instruction that was just executed.

Architecture Overview

Flags



Name	Symbol	Bit	Use
Carry	CF	0	Used to indicate if the previous operation resulted in a carry.
Parity	PF	2	Used to indicate if the last byte has an even number of 1's (i.e., even parity).
Adjust	AF	4	Used to support Binary Coded Decimal operations.
Zero	ZF	6	Used to indicate if the previous operation resulted in a zero result.
Sign	SF	7	Used to indicate if the result of the previous operation resulted in a 1 in the most significant bit (indicating negative in the context of signed data).
Direction	DF	10	Used to specify the direction (increment or decrement) for some string operations.
Overflow	OF	11	Used to indicate if the previous operation resulted in an overflow.

Memory

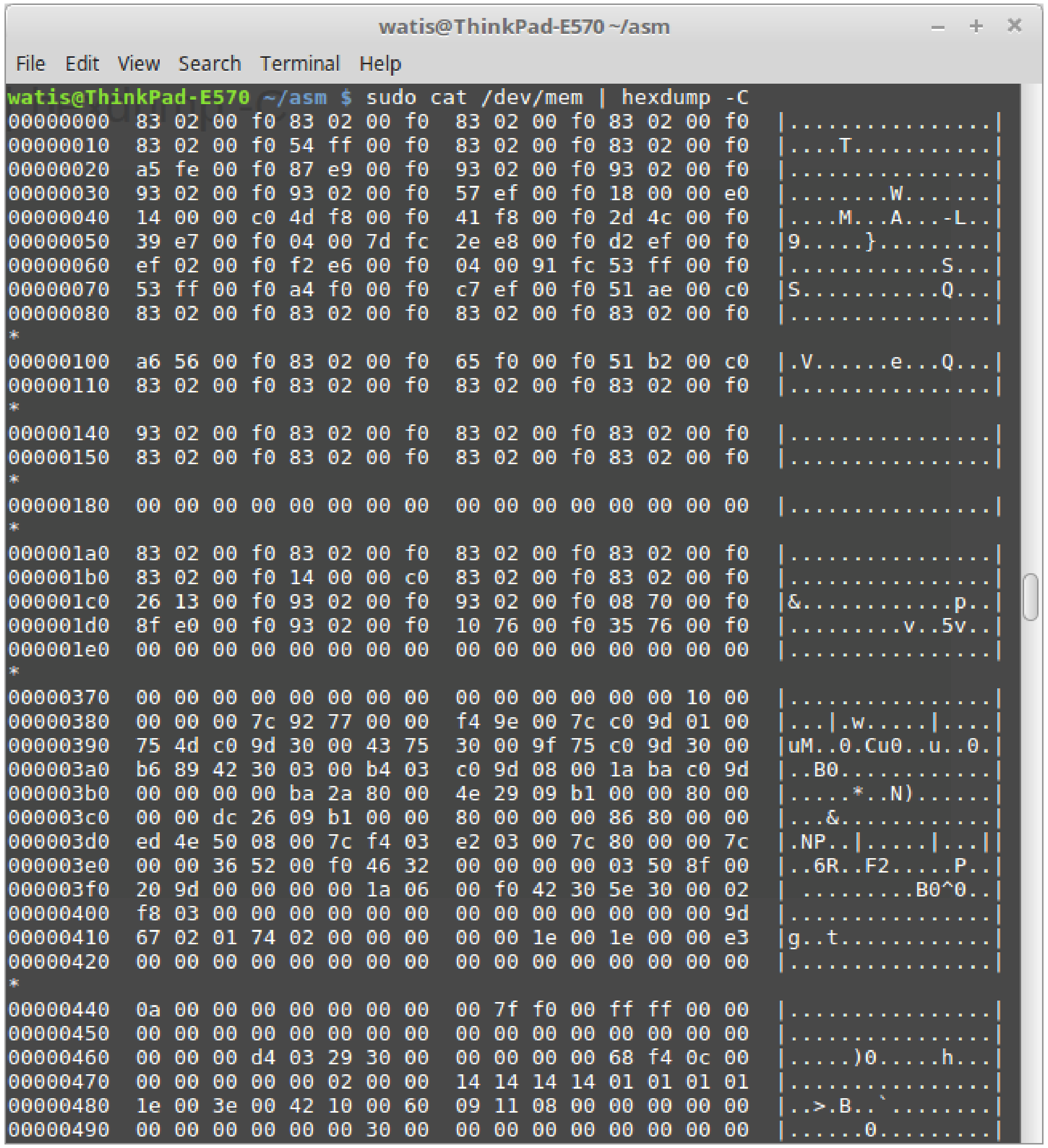
Primary storage (RAM: Random Access Memory)

Content	**Address**
7E	00000000
F2	00000001
23	00000002
14	00000003
⋮	⋮
⋮	⋮
⋮	⋮
C6	FFFFFFFE
AF	FFFFFFFF

Memory

Dumping memory

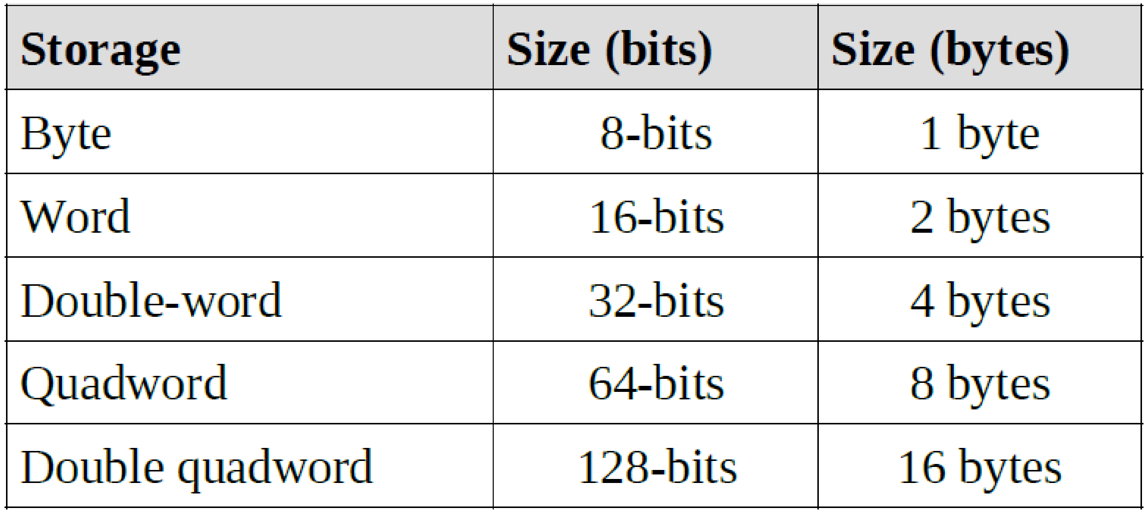
sudo cat /dev/mem | hexdump -C



```
watis@ThinkPad-E570 ~/asm $ sudo cat /dev/mem | hexdump -C
00000000  83 02 00 f0 83 02 00 f0  83 02 00 f0 83 02 00 f0  |................|
00000010  83 02 00 f0 54 ff 00 f0  83 02 00 f0 83 02 00 f0  |....T...........|
00000020  a5 fe 00 f0 87 e9 00 f0  93 02 00 f0 93 02 00 f0  |................|
00000030  93 02 00 f0 93 02 00 f0  57 ef 00 f0 18 00 00 e0  |........W.......|
00000040  14 00 00 c0 4d f8 00 f0  41 f8 00 f0 2d 4c 00 f0  |....M...A...-L..|
00000050  39 e7 00 f0 04 00 7d fc  2e e8 00 f0 d2 ef 00 f0  |9.....}.........|
00000060  ef 02 00 f0 f2 e6 00 f0  04 00 91 fc 53 ff 00 f0  |............S...|
00000070  53 ff 00 f0 a4 f0 00 f0  c7 ef 00 f0 51 ae 00 c0  |S...........Q...|
00000080  83 02 00 f0 83 02 00 f0  83 02 00 f0 83 02 00 f0  |................|
*
00000100  a6 56 00 f0 83 02 00 f0  65 f0 00 f0 51 b2 00 c0  |.V......e...Q...|
00000110  83 02 00 f0 83 02 00 f0  83 02 00 f0 83 02 00 f0  |................|
*
00000140  93 02 00 f0 83 02 00 f0  83 02 00 f0 83 02 00 f0  |................|
00000150  83 02 00 f0 83 02 00 f0  83 02 00 f0 83 02 00 f0  |................|
*
00000180  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
000001a0  83 02 00 f0 83 02 00 f0  83 02 00 f0 83 02 00 f0  |................|
000001b0  83 02 00 f0 14 00 00 c0  83 02 00 f0 83 02 00 f0  |................|
000001c0  26 13 00 f0 93 02 00 f0  93 02 00 f0 08 70 00 f0  |&............p..|
000001d0  8f e0 00 f0 93 02 00 f0  10 76 00 f0 35 76 00 f0  |.........v..5v..|
000001e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000370  00 00 00 00 00 00 00 00  00 00 00 00 00 00 10 00  |................|
00000380  00 00 00 7c 92 77 00 00  f4 9e 00 7c c0 9d 01 00  |...|.w.....|....|
00000390  75 4d c0 9d 30 00 43 75  30 00 9f 75 c0 9d 30 00  |uM..0.Cu0..u..0.|
000003a0  b6 89 42 30 03 00 b4 03  c0 9d 08 00 1a ba c0 9d  |..B0............|
000003b0  00 00 00 00 ba 2a 80 00  4e 29 09 b1 00 00 80 00  |.....*..N)......|
000003c0  00 00 dc 26 09 b1 00 00  80 00 00 00 86 80 00 00  |...&............|
000003d0  ed 4e 50 08 00 7c f4 03  e2 03 00 7c 80 00 00 7c  |.NP..|.....|...||
000003e0  00 00 36 52 00 f0 46 32  00 00 00 00 03 50 8f 00  |..6R..F2.....P..|
000003f0  20 9d 00 00 00 00 1a 06  00 f0 42 30 5e 30 00 02  | .........B0^0..|
00000400  f8 03 00 00 00 00 00 00  00 00 00 00 00 00 00 9d  |................|
00000410  67 02 01 74 02 00 00 00  00 00 1e 00 1e 00 00 e3  |g..t............|
00000420  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000440  0a 00 00 00 00 00 00 00  00 7f f0 00 ff ff 00 00  |................|
00000450  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000460  00 00 00 d4 03 29 30 00  00 00 00 00 68 f4 0c 00  |.....)0.....h...|
00000470  00 00 00 00 00 02 00 00  14 14 14 14 01 01 01 01  |................|
00000480  1e 00 3e 00 42 10 00 60  09 11 08 00 00 00 00 00  |..>.B..`........|
00000490  00 00 00 00 00 00 30 00  00 00 00 00 00 00 00 00  |......0.........|
```

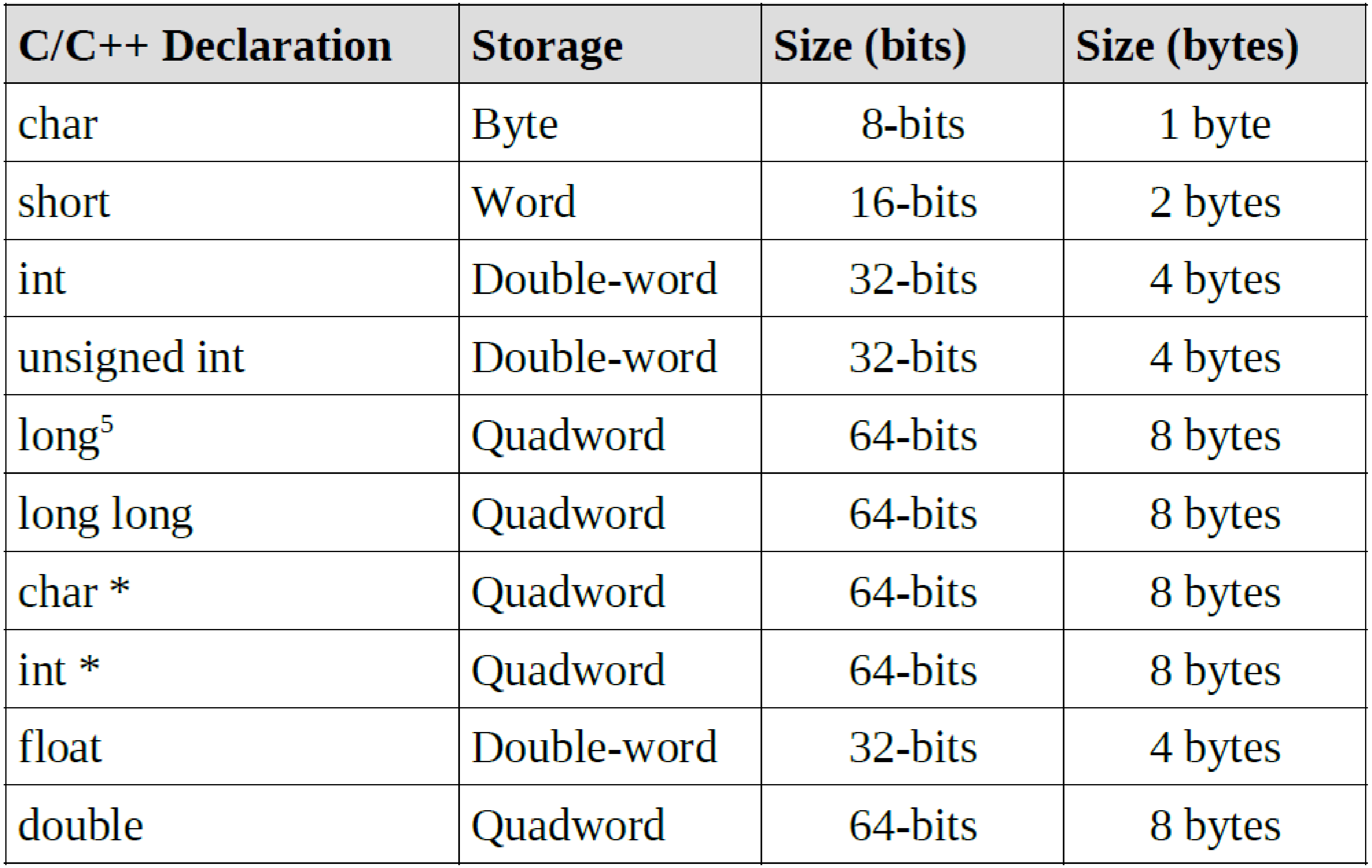
Data Storage Size

The x86-64 architecture supports a specific set of data storage size elements



Storage	Size (bits)	Size (bytes)
Byte	8-bits	1 byte
Word	16-bits	2 bytes
Double-word	32-bits	4 bytes
Quadword	64-bits	8 bytes
Double quadword	128-bits	16 bytes

Data Storage Size



C/C++ Declaration	Storage	Size (bits)	Size (bytes)
char	Byte	8-bits	1 byte
short	Word	16-bits	2 bytes
int	Double-word	32-bits	4 bytes
unsigned int	Double-word	32-bits	4 bytes
long[5]	Quadword	64-bits	8 bytes
long long	Quadword	64-bits	8 bytes
char *	Quadword	64-bits	8 bytes
int *	Quadword	64-bits	8 bytes
float	Double-word	32-bits	4 bytes
double	Quadword	64-bits	8 bytes

Memory Alignment (Endianness)

How multiple-byte data are stored in memory

Big endian: MSB is stored at low-address of memory
Little endian: LSB is stored at low-address of memory

Example: how the double word **7EF22314** is stored in memory

Content	Address
14	00000000
23	00000001
F2	00000002
7E	00000003
⋮	⋮
⋮	⋮
⋮	⋮
C6	FFFFFFFE
AF	FFFFFFFF

Little Endian

low-address Memory

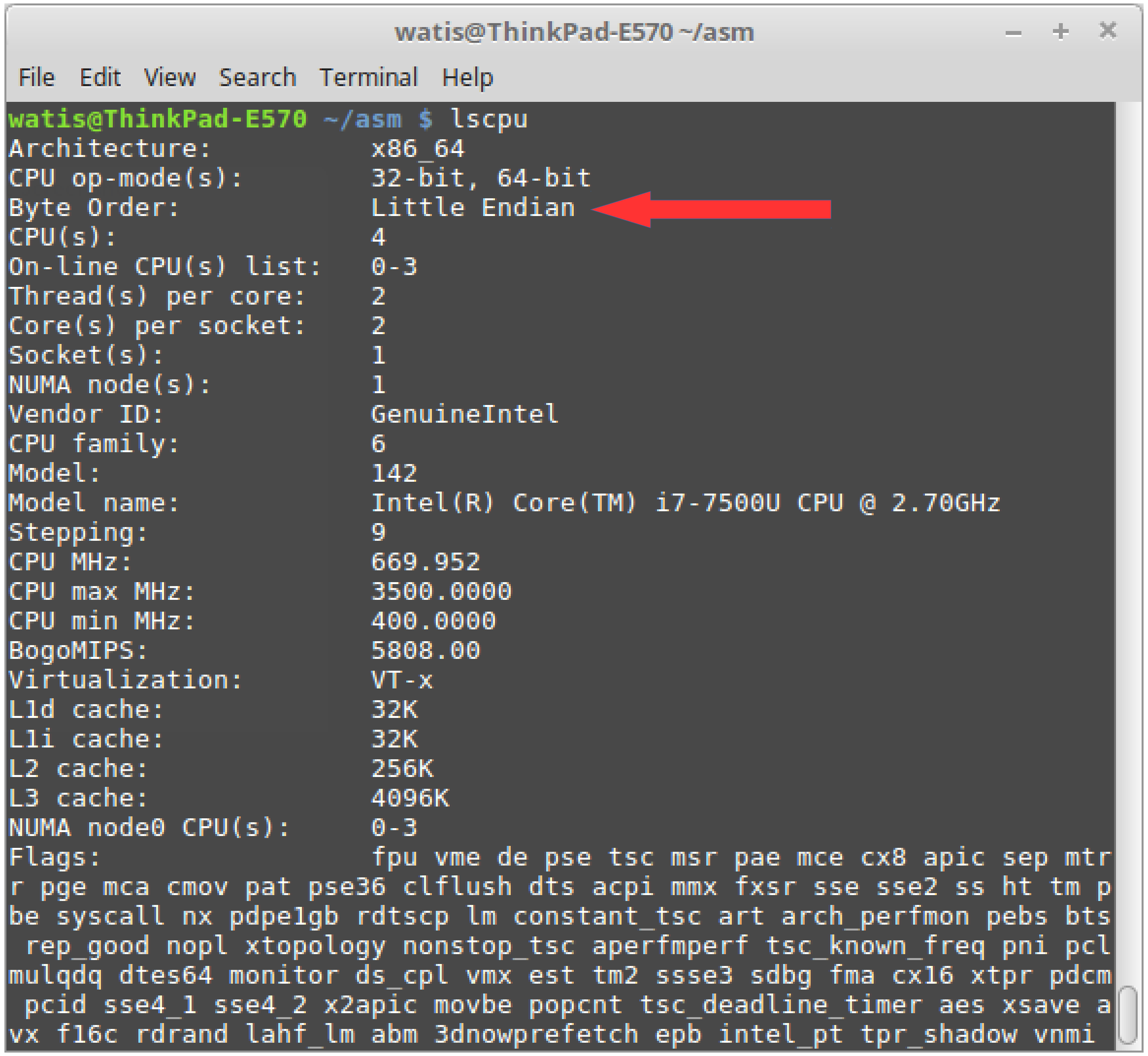
Content	Address
7E	00000000
F2	00000001
23	00000002
14	00000003
⋮	⋮
⋮	⋮
⋮	⋮
C6	FFFFFFFE
AF	FFFFFFFF

Big Endian

Memory Alignment

Checking memory alignment

lscpu


watis@ThinkPad-E570 ~/asm
File Edit View Search Terminal Help

```
watis@ThinkPad-E570 ~/asm $ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                4
On-line CPU(s) list:   0-3
Thread(s) per core:    2
Core(s) per socket:    2
Socket(s):             1
NUMA node(s):          1
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 142
Model name:            Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz
Stepping:              9
CPU MHz:               669.952
CPU max MHz:           3500.0000
CPU min MHz:           400.0000
BogoMIPS:              5808.00
Virtualization:        VT-x
L1d cache:             32K
L1i cache:             32K
L2 cache:              256K
L3 cache:              4096K
NUMA node0 CPU(s):     0-3
Flags:                 fpu vme de pse tsc msr pae mce cx8 apic sep mtr
r pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm p
be syscall nx pdpe1gb rdtscp lm constant_tsc art arch_perfmon pebs bts
 rep_good nopl xtopology nonstop_tsc aperfmperf tsc_known_freq pni pcl
mulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 sdbg fma cx16 xtpr pdcm
 pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave a
vx f16c rdrand lahf_lm abm 3dnowprefetch epb intel_pt tpr_shadow vnmi
```

Memory Layout

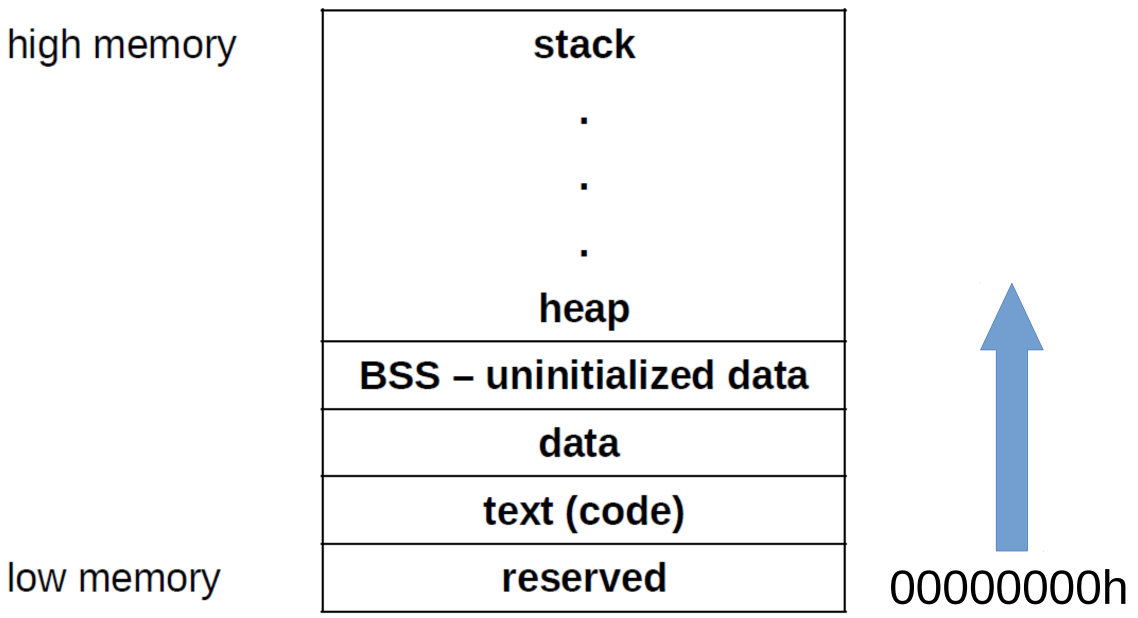
There are 2 major types of computer architecture:

- Von Neumann Architecture : Code & data **share** the same memory space
- Harvard Architecture : Code & data use **separate** memory spaces

X86-64

Reserved memory: system memory (BIOS) and OS→ NOT available to users

Text:	User programs	
Data:	Initialized variables	i.e., int x = 0;
BSS:	Uninitialized variables	i.e., int y;
Heap:	Dynamic variable allocation	i.e., malloc();
Stack:	Stack	


high memory
stack
.
.
.
heap
BSS – uninitialized data
data
text (code)
reserved
low memory
00000000h