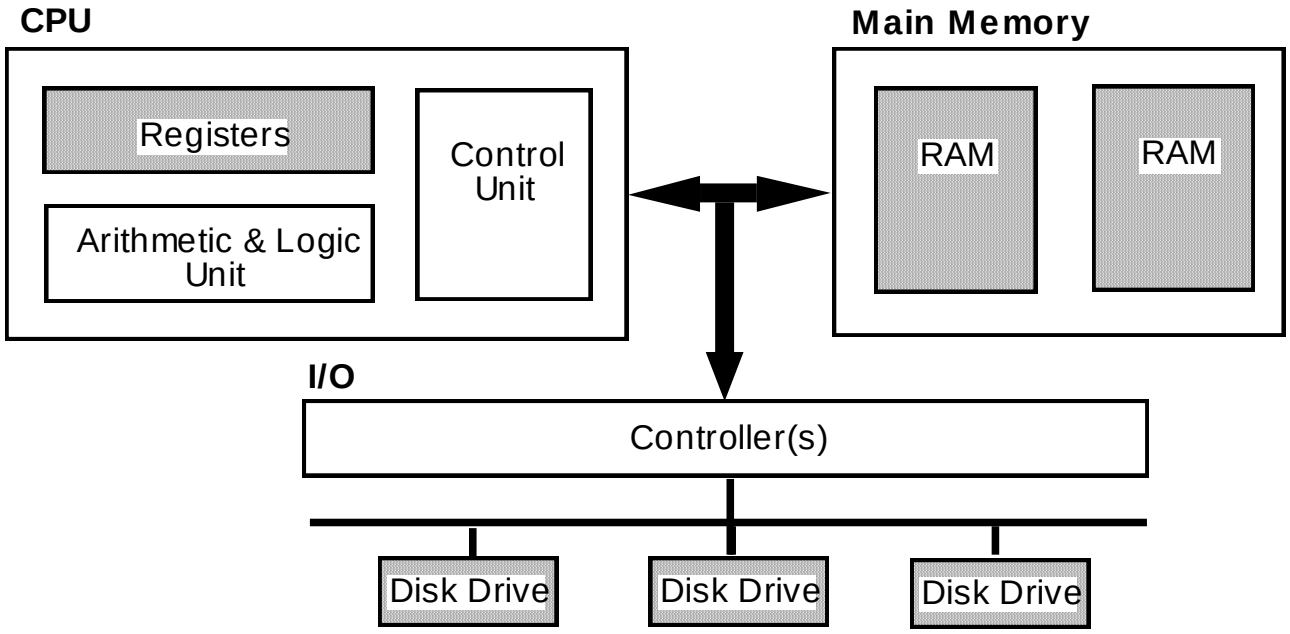
Main Memory (RAM) Organisation

Computers employ many different types of memory (semi-conductor, magnetic disks, USB sticks, DVDs etc.) to hold data and programs. Each type has its own characteristics and uses. We will look at the way that Main Memory (RAM) is organised and briefly at the characteristics of Register Memory and Disk Memory. Let us locate these 3 types of memory in a simplified model of a computer:



Register Memory

Registers are memories located within the Central Processing Unit (CPU). They are few in number (there are rarely more than 64 registers) and also small in size, typically a register is less than 64 bits; 32-bit and more recently 64-bit are common in desktops.

The contents of a register can be "read" or "written" very quickly[1] however, often an order of magnitude faster than main memory and several orders of magnitude faster than disk memory.

Different kinds of register are found within the CPU. General Purpose Registers[2] are available for general[3] use by the programmer. Unless the context implies otherwise we'll use the term "register" to refer to a General Purpose Register within the CPU. Most modern CPU's have between 16 and 64 general purpose registers. Special Purpose Registers have specific uses and are either non-programmable and internal to the CPU or accessed with special instructions by the programmer. Examples of such registers that we will encounter later in the course include: the Program Counter register (PC), the Instruction Register (IR), the ALU Input & Output registers, the Condition Code (Status/Flags) register, the Stack Pointer register (SP). The size (the number of bits in the register) of

1 e.g. less than a nanosecond (10^{-9} sec)
2 Occasionally called Working Registers
3 Used for performing calculations, moving and manipulating data etc.

these registers varies according to register type. The Word Size of an architecture is often (but not always!) defined by the size of the general purpose registers.

In contrast to main memory and disk memory, registers are referenced directly by specific instructions or by encoding a register number within a computer instruction. At the programming (assembly) language level of the CPU, registers are normally specified with special identifiers (e.g. R0, R1, R7, SP, PC).

As a final point, the contents of a register are lost if power to the CPU is turned off, so registers are unsuitable for holding long-term information or information that is needed for retention after a power-shutdown or failure. Registers are however, the fastest memories, and if exploited can result in programs that execute very quickly.

Main Memory (RAM)

If we were to sum all the bits of all registers within CPU, the total amount of memory probably would not exceed 5,000 bits. Most computational tasks undertaken by a computer require a lot more memory. Main memory is the next[4] fastest memory within a computer and is much larger in size. Typical main memory capacities for different kinds of computers are: PC 512MB[5], fileserver 4GB, database server 8GB. Computer architectures also impose an architectural constraint on the maximum allowable RAM. This constraint is normally equal to $\mathbf{2^{WordSize}}$ memory locations.

RAM[6] (Random[7] Access Memory) is the most common form of Main Memory. RAM is normally located on the motherboard and so is typically less than 12 inches from the CPU. ROM (Read Only Memory) is like RAM except that its contents cannot be overwritten and its contents are not lost if power is turned off (ROM is non-volatile).

Although slower than register memory, the contents of any location[8] in RAM can still be "read" or "written" very quickly[9]. The time to read or write is referred to as the **access time** and is constant for all RAM locations.

In contrast to register memory, RAM is used to hold both program code (instructions) and data (numbers, strings etc). Programs are "loaded" into RAM from a disk prior to execution by the CPU.

Locations in RAM are identified by an **addressing scheme** *e.g.* numbering the bytes in RAM from 0 onwards[10]. Like registers, the contents of RAM are lost if the power is turned off.

---

4 Actually many computers systems also include Cache memory, which is faster than Main memory, but slower than register memory. We will ignore Cache memories in this course.

5 $1K = 2^{10} = 1024$, $1M = 2^{20}$, $1G = 2^{30}$, 'B' will be used for Bytes, and 'b' or 'bit' for bits, cf. 1MB and 1Mbit

6 There are many types of RAM technologies.

7 Random is a Misnomer. Direct Access Memory would have been a better term.

8 Typically a byte multiple.

9 e.g. less than 10 nanoseconds (10×10^{-9} sec)

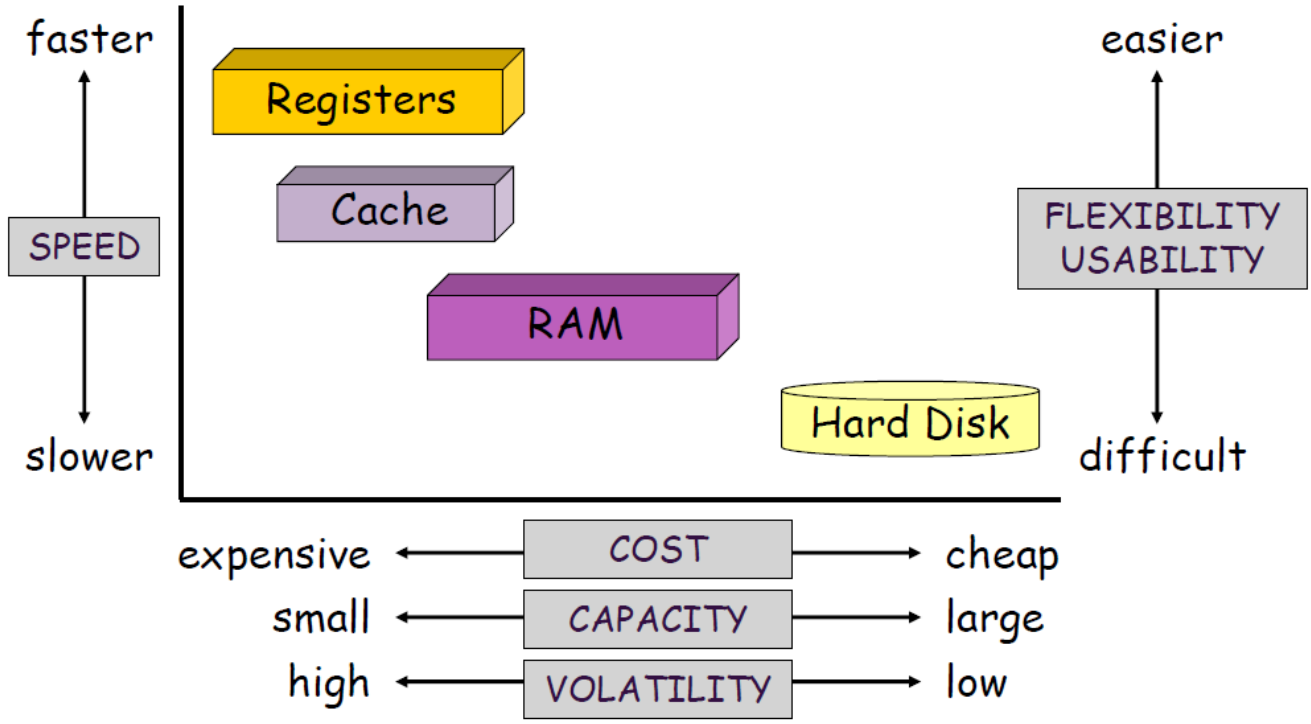
10 Some RAM locations (typically those with the lowest & highest addresses) may cause side-effects, e.g. cause data to be transferred to/from external devices

Disk Memory

Disk memory[11] is used to hold programs and data over the longer term. The **contents of a disk are NOT lost if the power is turned off.** Typical hard disk capacities range from 100GB to over 1TB ($1x10^{30}$). Disks are much slower than register and main memory, the access-time (known as the seek-time) to data on disk is typically between 2 and 4 milli-seconds, although disk drives can transfer thousands of bytes in one go achieving transfer rates from 25MB/s to 500MB/s.

Disks can be housed internally within a computer "box" or externally in an enclosure connected by a fast USB or firewire cable[12]. Disk locations are identified by special disk addressing schemes (e.g. track and sector numbers).

Summary of Characteristics



11 Some authors refer to disk memory as disk storage.

12 For details about how disks and other storage devices work, check out Tanenbaum or Stallings.

SRAM, DRAM, SDRAM, DDR SDRAM

There are many kinds of RAM and new ones are invented all the time. One aim is to make RAM access as fast as possible in order to keep up with the increasing speed of CPUs.

SRAM (Static RAM) is the fastest form of RAM but also the most expensive. Due to its cost it is not used as main memory but rather for cache memory. Each bit requires a 6-transistor circuit.

DRAM (Dynamic RAM) is not as fast as SRAM but is cheaper and is used for main memory. Each bit uses a single capacitor and single transistor circuit. Since capacitors lose their charge, DRAM needs to be refreshed every few milliseconds. The memory system does this transparently. There are many implementations of DRAM, two well-known ones are SDRAM and DDR SDRAM.

SDRAM (Synchronous DRAM) is a form of DRAM that is synchronised with the clock of the CPU's system bus, sometimes called the front-side bus (FSB). As an example, if the system bus operates at 167Mhz over an 8-byte (64-bit) data bus , then an SDRAM module could transfer 167 x 8 ~ 1.3GB/sec.

DDR SDRAM (Double-Data Rate SDRAM) is an optimisation of SDRAM that allows data to be transferred on both the rising edge and falling edge of a clock signal, effectively doubling the amount of data that can be transferred in a period of time. For example a PC-3200 DDR-SDRAM module operating at 200Mhz can transfer 200 x 8 x 2 ~ 3.2GB/sec over an 8-byte (64-bit) data bus. DDR3 continues the trend, doubling the minimum read or write unit to 8 consecutive words.

ROM, PROM, EPROM, EEPROM, Flash

In addition to RAM, they are also a range of other semi-conductor memories that retain their contents when the power supply is switched off.

ROM (Read Only Memory) is a form of semi-conductor that can be written to once, typically in bulk at a factory. ROM was used to store the "boot" or start-up program (so called firmware) that a computer executes when powered on, although it has now fallen out-of-favour to more flexible memories that support occasional writes. ROM is still used in systems with fixed functionalities, e.g. controllers in cars, household appliances etc.

PROM (Programmable ROM) is like ROM but allows end-users to write their own programs and data. It requires special PROM writing equipment. Note: users can only write-once to PROM.

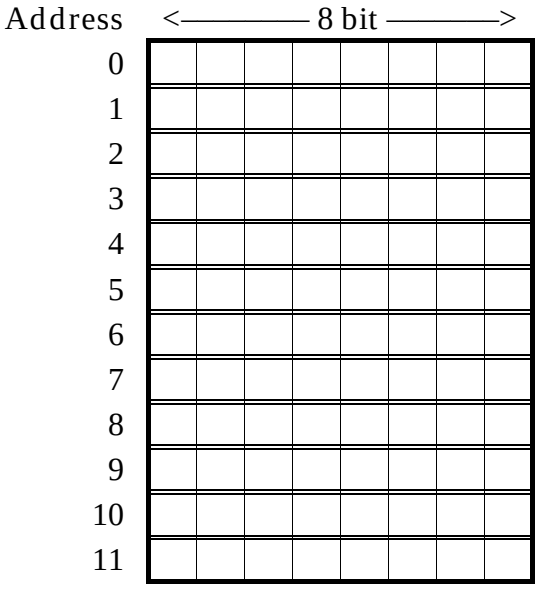
EPROM (Erasable PROM). With EPROM we can erase (using strong ultra-violet light) the contents of the chip and rewrite it with new contents, typically several thousand times. It is commonly used to store the "boot" program of a computer, known as the firmware. PCs call this firmware, the BIOS (Basic I/O System). Other systems use Open Firmware. Intel-based Macs use EFI (Extensible Firmware Interface).

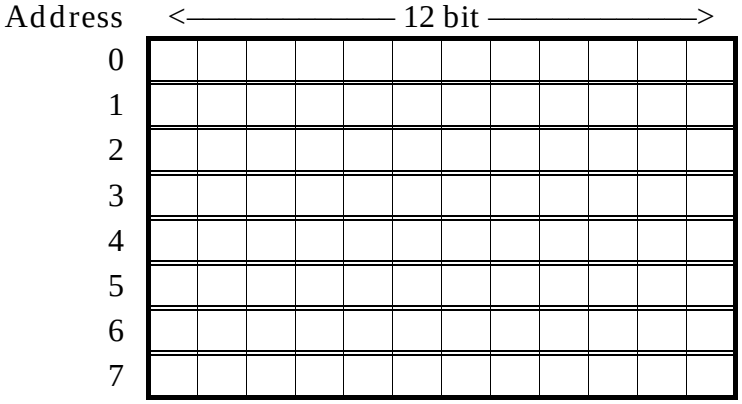
EEPROM (Electrically Erasable PROM). As the name implies the contents of EEPROMs are erased electrically. EEPROMSs are also limited to the number of erase-writes that can be performed (e.g, 100,000) but support updates (erase-writes) to individual bytes whereas EPROM updates the whole memory and only supports around 10,000 erase-write cycles.

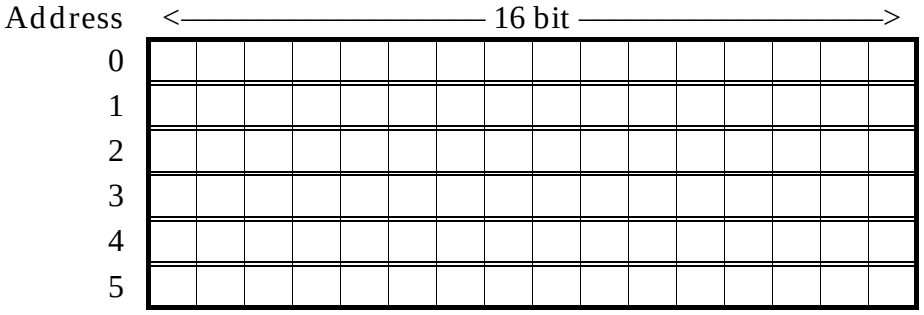
FLASH memory is a cheaper form of EEPROM where updates (erase-writes) can only be performed on blocks of memory, not on individual bytes. Flash memories are found in USB sticks, flash cards and typically range in size from 1GB to 32GB. The number of erase/write cycles to a block is typically several hundred thousand before the block can no longer be written.

Main Memory Organisation

Main memory can be considered to be organised as a matrix of bits. Each row represents a memory location, the number of bits in which is often the word size of the architecture, although it can be a word multiple (e.g. two words) or a partial word (e.g. half word**). For simplicity we will assume that data within main memory can only be read or written a single row (memory location) at a time.** For a 96-bit memory we could organise the memory as 12x8 bits, or 8x12 bits or, 6x16 bits, or even as 96x1 bits or 1x96 bits. Each row also has a natural number called its **address** which is used for selecting the row:

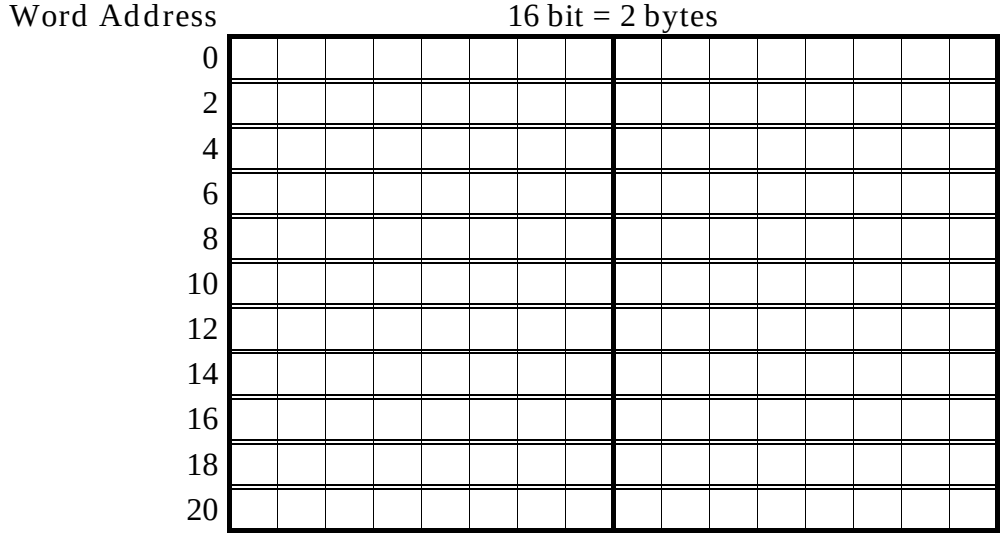






Byte Addressing

Main-memories generally store and recall rows, which are multi-byte in length (e.g. 16-bit word = 2 bytes, 32-bit word = 4 bytes). Many architectures, however, make main memory **byte-addressable** rather than **word addressable**. In such architectures the CPU and/or the main memory hardware is capable of reading/writing any individual byte. Here is an example of a main memory with 16-bit memory locations[13]. Note how the memory locations (rows) have even addresses.



Byte Ordering

The ordering of bytes within a **multi-byte** data item defines the endian-ness of the architecture.

In BIG-ENDIAN systems the most significant byte of a multi-byte data item always has the lowest address, while the least significant byte has the highest address.

In LITTLE-ENDIAN systems, the least significant byte of a multi-byte data item always has the lowest address, while the most significant byte has the highest address.

In the following example, table cells represent bytes, and the cell numbers indicate the address of that byte in main memory. Note: by convention we draw the bytes within a memory word left-to-right for big-endian systems, and right-to-left for little-endian systems.

Big-Endian

Word Address				
0	0	1	2	3
4	4	5	6	7
8	8	9	10	11
12	12	13	14	15

Little-Endian

Word Address				
0	3	2	1	0
4	7	6	5	4
8	11	10	9	8
12	15	14	13	12

[13] To avoid confusion we will use the term **memory word** for a word-sized memory location.

MSB ————————-> LSB MSB ————————> LSB

Note: an N-character ASCII string value is not treated as one large multi-byte value, but rather as N byte values, i.e. the first character of the string always has the lowest address, the last character has the highest address. This is true for both big-endian and little-endian. An N-character Unicode string would be treated as N two-byte values and each two-byte value would require suitable byte-ordering.

Example: Show the contents of memory at word address 24 if that word holds the number given by 122E 5F01H in both the big-endian and the little-endian schemes?

Big Endian

MSB ————————> LSB

	24	25	26	27
Word 24	12	2E	5F	01

Little Endian

MSB ————————> LSB

	27	26	25	24
Word 24	12	2E	5F	01

Example: Show the contents of main memory from word address 24 if those words hold the ASCII string value JIM SMITH.

Big Endian

	+0	+1	+2	+3
Word 24	J	I	M	
Word 28	S	M	I	T
Word 32	H	?	?	?

Little Endian

	+3	+2	+1	+0
Word 24		M	I	J
Word 28	T	I	M	S
Word 32	?	?	?	H

The bytes labelled with ? are unknown. They could hold important data, or they could be don't care bytes – the interpretation is left up to the programmer.

Unfortunately computer systems[14], in use today are split between those that are big-endian, and those that are little-endian[15]. This leads to problems when a big-endian computer wants to transfer data to a little-endian computer. Some architectures, for example the PowerPC and ARM, allow the endian-ness of the architecture to be changed programmatically.

Word Alignment

Although main-memories are generally organised as byte-addressed rows of words and accessed a row at a time, some architectures, allow the CPU to access any word-sized bit-group regardless of its byte address. We say that accesses that begin on a memory word boundary are **aligned accesses** while accesses that do not begin on word boundaries are **unaligned accesses**.

[14] The interested student might want to read the paper, "On Holy Wars and a Plea for Peace", D. Cohen, IEEE Computer, Vol 14, Pages 48-54, October 1981.

[15] The Motorola 68000 architecture is big-endian, while the Intel Pentium architecture is little-endian.

Address	Memory (16-bit) word	
0	**MSB**	**LSB**
2		
4		**MSB**
6	**LSB**	

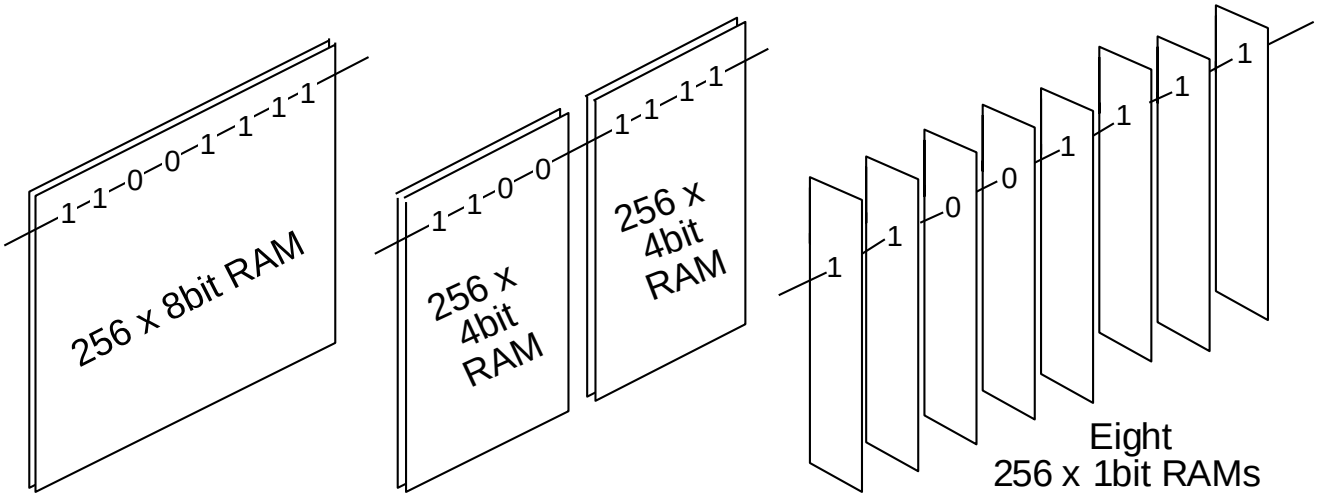
Word starting at Address 0 is Aligned

Word starting at Address 5 is Unaligned

Reading an unaligned word from RAM requires (i) reading of adjacent words, (ii) selecting the required bytes from each word and (iii) concatenating those bytes together => SLOW. Writing an unaligned word is more complex and slower[16]. For this reason some architectures prohibit unaligned word accesses. e.g. on the 68000 architecture, words must not be accessed starting from an odd-address (e.g. 1, 3, 5, 7 etc), on the SPARC architecture, 64-bit data items must have a byte address that is a multiple of 8.

Memory Modules, Memory Chips

So far, we have looked at the logical organisation of main memory. Physically RAM comes on small memory modules (little green printed circuit-boards about the size of a finger). A typical memory module holds 512MB to 2GB. The computer's motherboard will have slots to hold 2, 4 maybe 8 memory modules. Each memory module is itself comprised of several memory chips. For example here are 3 ways of forming a 256x8 bit memory module.



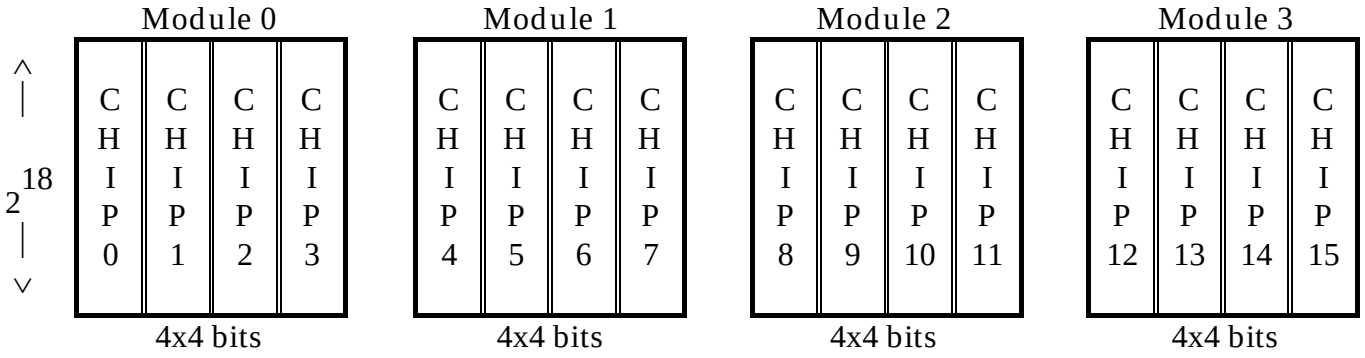
In the first case, main memory is built with a single memory chip. In the second, we use two memory chips, one gives us the most significant 4 bits, the other, the least significant 4 bits. In the third we use 8 memory chips, each chip gives us 1 bit - to read an 8 bit memory word, we would have to access all 8 memory chips simultaneously and concatenate the bits.

On PCs, memory modules are known as DIMMs (dual inline memory modules) and support 64-bit transfers. Previous generation of modules were called SIMMs (single inline memory modules) which supported 32-bit data transfers.

16 Describe a method for doing an unaligned word write operation.

Example: Given Main Memory = 1M x 16 bit (word addressable),

RAM chips = 256K x 4 bit



$$\text{RAM chips per memory module} = \frac{\text{Width of Memory Word}}{\text{Width of RAM Chip}} = 16/4 = 4$$

18 bits are required to address a RAM chip (since 256K = 2^{18} = Length of RAM Chip)

A 1Mx16 bit word-addressed memory requires 20 address bits (since 1M =2^{20})

Therefore 2 bits (=20–18) are needed to select a module.

The total number of RAM Chips = (1M x 16) / (256K x 4) = 16.

Total number of Modules = Total number of RAM chips / RamChipsPerModule = 16/4 = 4

Interleaved Memory

When memory consists of several memory modules, some address bits will select the module, and the remaining bits will select a row within the selected module.

When the module selection bits are the least significant bits of the memory address, we call the resulting memory a **low-order interleaved** memory.

When the module selection bits are the most significant bits of the memory address, we call the resulting memory a **high-order interleaved** memory.

Interleaved memory can yield performance advantages **if** more than one memory module can be read/written at a time:

(i) for low-order interleave, we can read the same row in each module. This is good for a single multi-word access of sequential data such as program instructions, or elements in a vector,

(ii) for high-order interleave, different modules can be independently accessed by different units. This is good if the CPU can access rows in one module, while at the same time, the hard disk (or a second CPU) can access different rows in another module concurrently.

Example: Given that Main Memory = 1Mx8bits, RAM chips = 256K x 4bit. For this memory we would require 4x2=8 RAM chips. Each chip would require 18 address bits (ie. 2^{18} = 256K) and the full 1Mx16 bit memory would require 20 address bits (ie. 2^{20} = 1M).