WIKIPEDIA

Word (computer architecture)

Bit
1 2 4 8 12 16 18 24 26 31 32 36 48 60 64 128 256 512
Application
8 16 32 64
Binary floating point precision
16 32 40 64 80 128 256 ×½ ×1 ×2 ×4 ×8
Decimal floating point precision
32 64 128

In computing, a **word** is the natural unit of data used by a particular processor design. A word is a fixed-sized piece of data handled as a unit by the instruction set or the hardware of the processor. The number of bits in a word (the *word size*, *word width*, or *word length*) is an important characteristic of any specific processor design or computer architecture.

The size of a word is reflected in many aspects of a computer's structure and operation; the majority of the registers in a processor are usually word sized and the largest piece of data that can be transferred to and from the working memory in a single operation is a word in many (not all) architectures. The largest possible address size, used to designate a location in memory, is typically a hardware word (here, "hardware word" means the full-sized natural word of the processor, as opposed to any other definition used).

Modern processors, including embedded systems, usually have a word size of 8, 16, 24, 32, or 64 bits, while modern general purpose computers usually use 32 or 64 bits. Special purpose digital processors, such as DSPs for instance, may use other sizes, and many other sizes have been used historically, including 9, 12,[1] 18, 24, 26, 36, 39, 40, 48, and 60 bits. The slab is an example of a system with an earlier word size. Several of the earliest computers (and a few modern as well) used BCD rather than plain binary, typically having a word size of 10 or 12 decimal digits, and some early decimal computers had no fixed word length at all.

The size of a word can sometimes differ from the expected due to backward compatibility with earlier computers. If multiple compatible variations or a family of processors share a common architecture and instruction set but differ in their word sizes, their documentation and software may become notationally complex to accommodate the difference (see Size families below).

Contents

Uses of words

Depending on how a computer is organized, word-size units may be used for:

Fixed point numbers
Holders for fixed point, usually integer, numerical values may be available in one or in several different sizes, but one of the sizes available will almost always be the word. The other sizes, if any, are likely to be multiples or fractions of the word size. The smaller sizes are normally used only for efficient use of memory; when loaded into the processor, their values usually go into a larger, word sized holder.

Floating point numbers
Holders for floating point numerical values are typically either a word or a multiple of a word.

Addresses
Holders for memory addresses must be of a size capable of expressing the needed range of values but not be excessively large, so often the size used is the word though it can also be a multiple or fraction of the word size.

Registers
Processor registers are designed with a size appropriate for the type of data they hold, e.g. integers, floating point numbers or addresses. Many computer architectures use "general purpose registers" that can hold any of several types of data, these registers must be sized to hold the largest of the types, historically this is the word size of the architecture though increasingly special purpose, larger, registers have been added to deal with newer types.

Memory–processor transfer
When the processor reads from the memory subsystem into a register or writes a register's value to memory, the amount of data transferred is often a word. Historically, this amount of bits which could be transferred in one cycle was also called a *catena* in some environments (such as the Bull GAMMA 60).[2][3] In simple memory subsystems, the word is transferred over the memory data bus, which typically has a width of a word or half-word. In memory subsystems that use caches, the word-sized transfer is the one between the processor and the first level of cache; at lower levels of the memory hierarchy larger transfers (which are a multiple of the word size) are normally used.

Unit of address resolution
In a given architecture, successive address values designate successive units of memory; this unit is the unit of address resolution. In most computers, the unit is either a character (e.g. a byte) or a word. (A few computers have used bit resolution.) If the unit is a word, then a larger amount of memory can be accessed using an address of a given size at the cost of added complexity to access individual characters. On the other hand, if the unit is a byte, then individual characters can be addressed (i.e. selected during the memory operation).

Instructions
Machine instructions are normally the size of the architecture's word, such as in RISC

architectures, or a multiple of the "char" size that is a fraction of it. This is a natural choice since instructions and data usually share the same memory subsystem. In Harvard architectures the word sizes of instructions and data need not be related, as instructions and data are stored in different memories; for example, the processor in the 1ESS electronic telephone switch had 37-bit instructions and 23-bit data words.

Word size choice

When a computer architecture is designed, the choice of a word size is of substantial importance. There are design considerations which encourage particular bit-group sizes for particular uses (e.g. for addresses), and these considerations point to different sizes for different uses. However, considerations of economy in design strongly push for one size, or a very few sizes related by multiples or fractions (submultiples) to a primary size. That preferred size becomes the word size of the architecture.

Character size was in the past (pre-variable-sized character encoding) one of the influences on unit of address resolution and the choice of word size. Before the mid-1960s, characters were most often stored in six bits; this allowed no more than 64 characters, so the alphabet was limited to upper case. Since it is efficient in time and space to have the word size be a multiple of the character size, word sizes in this period were usually multiples of 6 bits (in binary machines). A common choice then was the 36-bit word, which is also a good size for the numeric properties of a floating point format.

After the introduction of the IBM System/360 design, which used eight-bit characters and supported lower-case letters, the standard size of a character (or more accurately, a byte) became eight bits. Word sizes thereafter were naturally multiples of eight bits, with 16, 32, and 64 bits being commonly used.

Variable word architectures

Early machine designs included some that used what is often termed a *variable word length*. In this type of organization, a numeric operand had no fixed length but rather its end was detected when a character with a special marking, often called word mark, was encountered. Such machines often used binary coded decimal for numbers. This class of machines included the IBM 702, IBM 705, IBM 7080, IBM 7010, UNIVAC 1050, IBM 1401, and IBM 1620.

Most of these machines work on one unit of memory at a time and since each instruction or datum is several units long, each instruction takes several cycles just to access memory. These machines are often quite slow because of this. For example, instruction fetches on an IBM 1620 Model I take 8 cycles just to read the 12 digits of the instruction (the Model II reduced this to 6 cycles, or 4 cycles if the instruction did not need both address fields). Instruction execution took a completely variable number of cycles, depending on the size of the operands.

Word and byte addressing

The memory model of an architecture is strongly influenced by the word size. In particular, the resolution of a memory address, that is, the smallest unit that can be designated by an address, has often been chosen to be the word. In this approach, address values which differ by one designate adjacent memory words. This is natural in machines which deal

almost always in word (or multiple-word) units, and has the advantage of allowing instructions to use minimally sized fields to contain addresses, which can permit a smaller instruction size or a larger variety of instructions.

When byte processing is to be a significant part of the workload, it is usually more advantageous to use the byte, rather than the word, as the unit of address resolution. This allows an arbitrary character within a character string to be addressed straightforwardly. A word can still be addressed, but the address to be used requires a few more bits than the word-resolution alternative. The word size needs to be an integer multiple of the character size in this organization. This addressing approach was used in the IBM 360, and has been the most common approach in machines designed since then.

Individual bytes can be accessed on a word-oriented machine in one of two ways. Bytes can be manipulated by a combination of shift and mask operations in registers. Moving a single byte from one arbitrary location to another may require the equivalent of the following:

- LOAD the word containing the source byte
- SHIFT the source word to align the desired byte to the correct position in the target word
- AND the source word with a mask to zero out all but the desired bits
- LOAD the word containing the target byte
- AND the target word with a mask to zero out the target byte
- OR the registers containing the source and target words to insert the source byte
- STORE the result back in the target location

Alternatively many word-oriented machines implement byte operations with instructions using special *byte pointers* in registers or memory. For an example the PDP-10 byte pointer contained the size of the byte in bits (allowing different-sized bytes to be accessed), the bit position of the byte within the word, and the word address of the data. Instructions could automatically adjust the pointer to the next byte on, for example, load and deposit (store) operations.

Powers of two

Different amounts of memory are used to store data values with different degrees of precision. The commonly used sizes are usually a power of two multiple of the unit of address resolution (byte or word). Converting the index of an item in an array into the address of the item then requires only a shift operation rather than a multiplication. In some cases this relationship can also avoid the use of division operations. As a result, most modern computer designs have word sizes (and other operand sizes) that are a power of two times the size of a byte.

Size families

As computer designs have grown more complex, the central importance of a single word size to an architecture has decreased. Although more capable hardware can use a wider variety of sizes of data, market forces exert pressure to maintain backward compatibility while extending processor capability. As a result, what might have been the central word size in a fresh design has to coexist as an alternative size to the original word size in a backward compatible design. The original word size remains available in future designs, forming the basis of a size family.

In the mid-1970s, DEC designed the VAX to be a successor of the PDP-11. They used *word* for a 16-bit quantity, while *longword* referred to a 32-bit quantity. This was in contrast to earlier machines, where the natural unit of addressing memory would be called a *word*, while a quantity that is one half a word would be called a *halfword*. In fitting with this scheme, a VAX *quadword* is 64 bits.

Another example is the x86 family, of which processors of three different word lengths (16-bit, later 32- and 64-bit) have been released. As software is routinely ported from one word-length to the next, some APIs and documentation define or refer to an older (and thus shorter) word-length than the full word length on the CPU that software may be compiled for. Also, similar to how bytes are used for small numbers in many programs, a shorter word (16 or 32 bits) may be used in contexts where the range of a wider word is not needed (especially where this can save considerable stack space or cache memory space). For example, Microsoft's Windows API maintains the programming language definition of *WORD* as 16 bits, despite the fact that the API may be used on a 32- or 64-bit x86 processor, where the standard word size would be 32 or 64 bits, respectively. Data structures containing such different sized words refer to them as *WORD* (16 bits/2 bytes), *DWORD* (32 bits/4 bytes) and *QWORD* (64 bits/8 bytes) respectively. A similar phenomenon has developed in Intel's x86 assembly language – because of the support for various sizes (and backward compatibility) in the instruction set, some instruction mnemonics carry "d" or "q" identifiers denoting "double-", "quad-" or "double-quad-", which are in terms of the architecture's original 16-bit word size.

In general, new processors must use the same data word lengths and virtual address widths as an older processor to have binary compatibility with that older processor.

Often carefully written source code – written with source code compatibility and software portability in mind – can be recompiled to run on a variety of processors, even ones with different data word lengths or different address widths or both.

Table of word sizes

key: **b: bits, d: decimal digits, *w*: word size of architecture, *n*: variable size**							
Year	**Computer architecture**	**Word size *w***	**Integer sizes**	**Floating-point sizes**	**Instruction sizes**	**Unit of address resolution**	**Char size**
1837	Babbage Analytical engine	50 d	*w*	—	Five different cards were used for different functions, exact size of cards not known.	*w*	—
1941	Zuse Z3	22 b	—	*w*	8 b	*w*	—
1942	ABC	50 b	*w*	—	—	—	—
1944	Harvard Mark I	23 d	*w*	—	24 b	—	—
1946 (1948)	ENIAC (w/Panel #16[4])	10 d	*w*, 2*w* (w)	—	— (2 d, 4 d, 6 d, 8 d)	— —	—

{1953}	{w/Panel #26[5]}		{w}		{2 d, 4 d, 6 d, 8 d}	{w}	
1951	UNIVAC I	12 d	*w*	—	½*w*	*w*	1 d
1952	IAS machine	40 b	*w*	—	½*w*	*w*	5 b
1952	Fast Universal Digital Computer M-2	34 b	*w?*	*w*	34 b = 4 b opcode plus 3×10 b address	10 b	—
1952	IBM 701	36 b	½*w*, *w*	—	½*w*	½*w*, *w*	6 b
1952	UNIVAC 60	*n* d	1 d, ... 10 d	—	—	—	2 d, 3 d
1952	ARRA I	30 b	*w*	—	*w*	*w*	5 b
1953	IBM 702	*n* d	0 d, ... 511 d	—	5 d	d	1 d
1953	UNIVAC 120	*n* d	1 d, ... 10 d	—	—	—	2 d, 3 d
1953	ARRA II	30 b	*w*	2*w*	½*w*	*w*	5 b
1954 (1955)	IBM 650 (w/IBM 653)	10 d	*w*	— (w)	*w*	*w*	2 d
1954	IBM 704	36 b	*w*	*w*	*w*	*w*	6 b
1954	IBM 705	*n* d	0 d, ... 255 d	—	5 d	d	1 d
1954	IBM NORC	16 d	*w*	*w*, 2*w*	*w*	*w*	—
1956	IBM 305	*n* d	1 d, ... 100 d	—	10 d	d	1 d
1956	ARMAC	34 b	*w*	*w*	½*w*	*w*	5 b, 6 b
1957	Autonetics Recomp I	40 b	*w*, 79 b, 8 d, 15 d	—	½*w*	½*w*, *w*	5 b
1958	UNIVAC II	12 d	*w*	—	½*w*	*w*	1 d
1958	SAGE	32 b	½*w*	—	*w*	*w*	6 b
1958	Autonetics Recomp II	40 b	*w*, 79 b, 8 d, 15 d	2*w*	½*w*	½*w*, *w*	5 b
1958	Setun	6 trit (~9.5 b)	up to 6 tryte		up to 3 trytes		4 trit[?]
1958	Electrologica X1	27 b	*w*	2*w*	*w*	*w*	5 b, 6 b
1959	IBM 1401	*n* d	1 d, ...	—	1 d, 2 d, 4 d, 5 d, 7 d, 8 d	d	1 d
1959 (TBD)	IBM 1620	*n* d	2 d, ...	— (4 d, ... 102	12 d	d	2 d

				d)			
1960	LARC	12 d	*w*, 2*w*	*w*, 2*w*	*w*	*w*	2 d
1960	CDC 1604	48 b	*w*	*w*	½*w*	*w*	6 b
1960	IBM 1410	*n* d	1 d, ...	—	1 d, 2 d, 6 d, 7 d, 11 d, 12 d	d	1 d
1960	IBM 7070	10 d	*w*	*w*	*w*	*w*, d	2 d
1960	PDP-1	18 b	*w*	—	*w*	*w*	6 b
1960	Elliott 803	39 b + 1 parity					
1961	IBM 7030 (Stretch)	64 b	1 b, ... 64 b, 1 d, ... 16 d	*w*	½*w*, *w*	b, ½*w*, *w*	1 b, ... 8 b
1961	IBM 7080	*n* d	0 d, ... 255 d	—	5 d	d	1 d
1962	GE-6xx	36 b	*w*, 2 *w*	*w*, 2 *w*, 80 b	*w*	*w*	6 b, 9 b
1962	UNIVAC III	25 b	*w*, 2*w*, 3*w*, 4*w*, 6 d, 12 d	—	*w*	*w*	6 b
1962	Autonetics D-17B Minuteman I Guidance Computer	27 b	11 b, 24 b	—	24 b	*w*	—
1962	UNIVAC 1107	36 b	⅙*w*, ⅓*w*, ½*w*, *w*	*w*	*w*	*w*	6 b
1962	IBM 7010	*n* d	1 d, ...	—	1 d, 2 d, 6 d, 7 d, 11 d, 12 d	d	1 d
1962	IBM 7094	36 b	*w*	*w*, 2*w*	*w*	*w*	6 b
1963	Titan	48 b	*w*	*w*	*w*	*w*	*w*
1962	SDS 9 Series	24 b	*w*	2*w*	*w*	*w*	
1963/1966	PDP-6/PDP-10	36 b	*w*	*w*, 2 *w*	*w*	*w*	6 b, 9 b (typical)
1963	Gemini Guidance Computer	39 b	26 b	—	13 b	13 b, 26 b	—
1963 (1966)	Apollo Guidance Computer	15 b	*w*	—	*w*, 2*w*	*w*	—
	Saturn Launch						

1963	Vehicle Digital Computer	26 b	*w*	—	13 b	*w*	—
1964	CDC 6600	60 b	*w*	*w*	¼*w*, ½*w*	*w*	6 b
1964	Autonetics D-37C Minuteman II Guidance Computer	27 b	11 b, 24 b	—	24 b	*w*	4 b, 5 b
1965	IBM 360	32 b	½*w*, *w*, 1 d, ... 16 d	*w*, 2*w*	½*w*, *w*, 1½*w*	8 b	8 b
1965	UNIVAC 1108	36 b	⅙*w*, ¼*w*, ⅓*w*, ½*w*, *w*, 2*w*	*w*, 2*w*	*w*	*w*	6 b, 9 b
1965	PDP-8	12 b	*w*	—	*w*	*w*	8 b
1965	Electrologica X8	27 b	*w*	2*w*	*w*	*w*	6 b, 7b
1966	SDS Sigma 7	32 b	½*w*, *w*	*w*, 2*w*	*w*	8 b	8 b
1970	PDP-11	16 b	*w*	2*w*, 4*w*	*w*, 2*w*, 3*w*	8 b	8 b
1971	Intel 4004	4 b	*w*, d	—	2*w*, 4*w*	*w*	—
1972	Intel 8008	8 b	*w*, 2 d	—	*w*, 2*w*, 3*w*	*w*	8 b
1972	Calcomp 900	9 b	*w*	—	*w*, 2*w*	*w*	8 b
1974	Intel 8080	8 b	*w*, 2*w*, 2 d	—	*w*, 2*w*, 3*w*	*w*	8 b
1975	ILLIAC IV	64 b	*w*	*w*, ½*w*	*w*	*w*	—
1975	Motorola 6800	8 b	*w*, 2 d	—	*w*, 2*w*, 3*w*	*w*	8 b
1975	MOS Tech. 6501 MOS Tech. 6502	8 b	*w*, 2 d	—	*w*, 2*w*, 3*w*	*w*	8 b
1976	Cray-1	64 b	24 b, *w*	*w*	¼*w*, ½*w*	*w*	8 b
1976	Zilog Z80	8 b	*w*, 2*w*, 2 d	—	*w*, 2*w*, 3*w*, 4*w*, 5*w*	*w*	8 b
1978 (1980)	16-bit x86 (Intel 8086) (w/floating point: Intel 8087)	16 b	½*w*, *w*, 2 d	— (2*w*, 4*w*, 5*w*, 17 d)	½*w*, *w*, ... 7*w*	8 b	8 b
1978	VAX	32 b	¼*w*, ½*w*, *w*, 1 d, ... 31 d, 1 b, ... 32 b	*w*, 2*w*	¼*w*, ... 14¼*w*	8 b	8 b
	Motorola						

1979 (1984)	68000 series (w/floating point)	32 b	¼*w*, ½*w*, *w*, 2 d	— (*w*, 2*w*, 2½*w*)	½*w*, *w*, ... 7½*w*	8 b	8 b
1985	IA-32 (Intel 80386) (w/floating point)	32 b	8 b, ½*w*, *w*	— (½*w*, *w*, 80 b)	8 b, ... 15 b	8 b	8 b
1985	ARMv1	32 b	¼*w*, *w*	—	*w*	8 b	8 b
1985	MIPS	32 b	¼*w*, ½*w*, *w*	*w*, 2*w*	*w*	8 b	8 b
1991	Cray C90	64 b	32 b, *w*	*w*	¼*w*, ½*w*, 48 b	*w*	8 b
1992	Alpha	64 b	8 b, ¼*w*, ½*w*, *w*	*w*, 2*w*	½*w*	8 b	8 b
1992	PowerPC	32 b	¼*w*, ½*w*, *w*	*w*, 2*w*	*w*	8 b	8 b
1996	ARMv4 (w/Thumb)	32 b	¼*w*, ½*w*, *w*	—	*w* (½*w*, *w*)	8 b	8 b
2001	IA-64	64 b	8 b, ¼*w*, ½*w*, *w*	½*w*, *w*	41 b	8 b	8 b
2001	ARMv6 (w/VFP)	32 b	8 b, ½*w*, *w*	— (w, 2w)	½*w*, *w*	8 b	8 b
2003	x86-64	64 b	8 b, ¼*w*, ½*w*, *w*	½*w*, *w*, 80 b	8 b, ... 15 b	8 b	8 b
2013	ARMv8-A	64 b	8 b, ¼*w*, ½*w*, *w*	½*w*, *w*	½*w*	8 b	8 b
Year	**Computer architecture**	**Word size *w***	**Integer sizes**	**Floating-point sizes**	**Instruction sizes**	**Unit of address resolution**	**Char size**
***key:* b: bits, d: decimal digits, *w*: word size of architecture, *n*: variable size**							

[6][7]

See also

- Integer (computer science)

References

1. Schneider, Carl (2013) [1970]. *Datenverarbeitungs-Lexikon* (https://books.google.com/books?id=3aN9BwAAQBAJ) [*Lexicon of information technology*] (in German) (softcover reprint of hardcover 1st ed.). Wiesbaden, Germany: Springer Fachmedien Wiesbaden GmbH / Betriebswirtschaftlicher Verlag Dr. Th. Gabler GmbH. pp. 201, 308. ISBN 978-3-409-31831-0. doi:10.1007/978-3-663-13618-7 (https://doi.org/10.1007%2F978-3-663-13618-7). Retrieved 2016-05-24. "*slab*, Abk. aus *syllable* = Silbe, die kleinste adressierbare Informationseinheit für 12 bit zur Übertragung von zwei Alphazeichen oder drei numerischen Zeichen. *(NCR)* […] Hardware: Datenstruktur: NCR 315-

100 / NCR 315-RMC; Wortlänge: Silbe; Bits: 12; Bytes: –; Dezimalziffern: 3; Zeichen: 2; Gleitkommadarstellung: fest verdrahtet; Mantisse: 4 Silben; Exponent: 1 Silbe (11 Stellen + 1 Vorzeichen) [*slab*, abbr. for *syllable* = syllable, smallest addressable information unit for 12 bits for the transfer of two alphabetical characters or three numerical characters. *(NCR)* […] Hardware: Data structure: NCR 315-100 / NCR 315-RMC; Word length: Syllable; Bits: 12; Bytes: –; Decimal digits: 3; Characters: 2; Floating point format: hard-wired; Significand: 4 syllables; Exponent: 1 syllable (11 digits + 1 prefix)]"

2. Dreyfus, Phillippe (1958-05-08) [1958-05-06], written at Los Angeles, California, USA, "System design of the Gamma 60" (https://www.computer.org/csdl/proceedings/afips/1958/5052/00/50520130.pdf) (PDF), *Proceedings of the May 6–8, 1958, Western Joint Computer Conference: Contrasts in Computers*, ACM, New York, NY, USA, pp. 130–133, IRE-ACM-AIEE '58 (Western), archived (https://web.archive.org/web/20170403224547/https://www.computer.org/csdl/proceedings/afips/1958/5052/00/50520130.pdf) (PDF) from the original on 2017-04-03, retrieved 2017-04-03, "[…] Internal data code is used: Quantitative (numerical) data are coded in a 4-bit decimal code; qualitative (alpha-numerical) data are coded in a 6-bit alphanumerical code. The internal instruction code means that the instructions are coded in straight binary code.
As to the internal information length, the information quantum is called a "catena," and it is composed of 24 bits representing either 6 decimal digits, or 4 alphanumerical characters. This quantum must contain a multiple of 4 and 6 bits to represent a whole number of decimal or alphanumeric characters. Twenty-four bits was found to be a good compromise between the minimum 12 bits, which would lead to a too-low transfer flow from a parallel readout core memory, and 36 bits or more, which was judged as too large an information quantum. The catena is to be considered as the equivalent of a character in variable word length machines, but it cannot be called so, as it may contain several characters. It is transferred in series to and from the main memory.
Not wanting to call a "quantum" a word, or a set of characters a letter, (a word is a word, and a quantum is something else), a new word was made, and it was called a "catena." It is an English word and exists in Webster's although it does not in French. Webster's definition of the word catena is, "a connected series;" therefore, a 24-bit information item. The word catena will be used hereafter.
The internal code, therefore, has been defined. Now what are the external data codes? These depend primarily upon the information handling device involved. The Gamma 60 is designed to handle information relevant to any binary coded structure. Thus an 80-column punched card is considered as a 960-bit information item; 12 rows multiplied by 80 columns equals 960 possible punches; is stored as an exact image in 960 magnetic cores of the main memory with 2 card columns occupying one catena. […]"
3. Blaauw, Gerrit Anne; Brooks, Jr., Frederick Phillips; Buchholz, Werner (1962), "4: Natural Data Units", in Buchholz, Werner, *Planning a Computer System – Project Stretch* (http://archive.computerhistory.org/resources/text/IBM/Stretch/pdfs/Buchholz_102636426.pdf) (PDF), McGraw-Hill Book Company, Inc. / The Maple Press Company, York, PA., pp. 39–40, LCCN 61-10466 (https://lccn.loc.gov/61-10466), archived (https://web.archive.org/web/20170403014651/http://archive.computerhistory.org/resources/text/IBM/Stretch/pdfs/Buchholz_102636426.pdf) (PDF) from the original on 2017-04-03, retrieved 2017-04-03, "[…] Terms used here to describe the structure imposed by the machine design, in addition to *bit*, are listed below.
Byte denotes a group of bits used to encode a character, or the number of bits transmitted in parallel to and from input-output units. A term other than *character* is used here because a given character may be represented in different applications by more than one code, and different codes may use different numbers of bits (i.e., different byte sizes). In input-output transmission the grouping of bits may be completely arbitrary and have no relation to actual characters. (The term is coined from *bite*, but respelled to avoid accidental mutation to *bit*.)
A *word* consists of the number of data bits transmitted in parallel from or to memory in one memory cycle. Word size is thus defined as a structural property of the memory. (The term *catena* was coined for this purpose by the designers of the Bull GAMMA 60 computer.)
Block refers to the number of words transmitted to or from an input-output unit in response to a single input-output instruction. Block size is a structural property of an input-output unit; it may have been fixed by the design or left to be

varied by the program. […]"

4. Clippinger, Richard F. (1948-09-29). "A Logical Coding System Applied to the ENIAC (Electronic Numerical Integrator and Computer)" (http://ftp.arl.mil/~mike/comphist/48eniac-coding/). Aberdeen Proving Ground, Maryland, US: Ballistic Research Laboratories. Report No. 673; Project No. TB3-0007 of the Research and Development Division, Ordnance Department. Retrieved 2017-04-05.
5. Clippinger, Richard F. (1948-09-29). "A Logical Coding System Applied to the ENIAC" (http://ftp.arl.mil/~mike/comphist/48eniac-coding/sec8.html). Aberdeen Proving Ground, Maryland, US: Ballistic Research Laboratories. Section VIII: Modified ENIAC. Retrieved 2017-04-05.
6. Blaauw, Gerrit Anne; Brooks, Jr., Frederick Phillips (1997). *Computer Architecture: Concepts and Evolution*. Addison-Wesley. ISBN 0-201-10557-8.
7. Ralston, Anthony; Reilly, Edwin D. (1993). *Encyclopedia of Computer Science* (3rd ed.). Van Nostrand Reinhold. ISBN 0-442-27679-6.
