



LIBRARY OF THE  
UNIVERSITY OF ILLINOIS  
AT URBANA-CHAMPAIGN

510.84

I26r

no.61-80

cop.3





Digitized by the Internet Archive  
in 2013

<http://archive.org/details/computerdesignst61doug>







UNIVERSITY OF ILLINOIS  
DIGITAL COMPUTER LABORATORY  
GRADUATE COLLEGE

This volume is bound without no. 62, 65, & 80

. 61

---

which is/are unavailable.

ACCESS TIME

AS





UNIVERSITY OF ILLINOIS  
DIGITAL COMPUTER LABORATORY  
GRADUATE COLLEGE

INTERNAL REPORT NO. 61

COMPUTER DESIGN STUDY: ACCESS TIME

BY

ALEXANDER S. DOUGLAS

1-10-55



510.87  
IL6+  
no. 61-80  
cop. 3

COMPUTER DESIGN STUDY: ACCESS TIME

A study is currently being undertaken with a view to designing a computer operating at speeds greater than those of the Illiac. A preliminary study has been made of how to minimize delay due to the time taken to obtain access to, or to replace the contents of the primary storage of a particular class of computers. By primary storage we mean that storage from which instructions are brought forward for decoding, and to and from which words are transferred to be operated upon, but not including any such storage which can be regarded as having zero access time. By zero-access storage we mean any registers, of any length, the time for access to change one of which can be ignored relative to the time taken for operation upon its contents. The zero-access storage is always deemed to include the instruction decoding register or registers, the sequence control register, any modulation registers, and the 'operational' registers of the arithmetic, logic, or control units (if these exist independently). In order to isolate, so far as possible, the discussion of access times from a discussion of circuitry, we assume (a) that the times taken to carry out operations between words held in zero-access storage is given, (b) that a word from the primary store must be transferred to a zero-access store before it can be operated on, (c) that access to any word in the primary store takes the same time as to any other word (isochronic access), (d) that only one word can be transferred from the primary store at a time, (e) that an instruction occupies the whole of one word in the primary store, but that this word may specify more than one operation, and the associated number-word may be treated as more than one number for operational purposes, (f) that simultaneous carrying out of certain operations may be permitted, and, finally, (g) that, for practical and economical reasons, the zero-access storage is limited to about sixteen full-size registers, including those provided for sequence control, instruction decoding, modulating and operational purposes.

Since the primary storage has isochronic access we may denote the access time by  $a$  units of time, where this time is deemed to include the time required to decode the address in the primary store contained in the instruction being currently obeyed. We may suppose that the time to carry out each of the  $n$  possible operations between the contents of the zero-access storage registers respectively is  $b_n$ . We examine first a scheme involving instructions



containing at most one reference to the primary store. The following method of operation can be set up, and we study how to economize in time:

Stage Operation	(1)	(2)
1.	Decode the primary store reference (if any), and transfer the word designated to or from the primary.	Begin carrying out operations involving only the zero-access registers.
2.	Decode the address of the next instruction (or advance the sequence word count) and transfer that instruction to the decoding register.	Complete the operations begun above and carry out operations on the word acquired from the primary store (if any).

- Notes (i) We assume that an instruction is set up for decoding prior to Stage 1.
- (ii) Any word disposed of by transfer to the primary store must be a word available in the zero-access store before Stage 1 begins.
- (iii) A preliminary stage may be included to modulate the instruction in the decoder if desired. The most rapid method of operation is to modulate the next instruction in accordance with specifications contained in the instruction being decoded. This method is open to the objection that it implies heavy interdependence between instructions which makes error diagnosis and correction more difficult.

Stages 1 (1) and 2 (1) each take  $\underline{a}$  units, while Stage 1 (2) takes  $\sum_n \alpha_n \underline{b}_n$ , and the Stage 2 (2) takes  $\sum_n \beta_n \underline{b}_n$ . Should the instruction not refer to the primary, then Stage 1 (1) is replaced by 2 (1) and Stage 2 is omitted. We distinguish three cases:

Case I.  $\underline{a} \ll \text{minimum} (\underline{b}_1, \dots, \underline{b}_n)$ . In this case the whole time of the computer will be occupied with computing. The arrangement of the zero-access storage will not have a dominant influence (except possibly upon the values of  $\underline{b}_1, \dots, \underline{b}_n$ ) in the total time taken. It is pertinent to consider in this case use of a code involving multiple addressing of the primary and this will be discussed subsequently. Some advantage can be obtained by minimizing the number of accesses involved (see Case II), but this disappears if  $\text{minimum} (\underline{b}_1, \dots, \underline{b}_n) \geq 2\underline{a}$ .





Case II.  $\underline{a} \sim \text{average } (b_1, \dots, b_n)$ . The principal time saving that can be effected in this case is to eliminate, as many times as possible, the operation of Stage 2. This implies providing a zero-access storage larger than normally required for a one-address system, and adjusting the instruction code so that this storage can be used as 'working space'. The amount of zero-access storage that is desirable depends upon the type of problem likely to be tackled and upon economic considerations. Experience would indicate that not less than four registers, exclusive of modulation, accumulation and operational storage, should be provided for 'working space' and that more are often desirable. In this case little or no advantage can be gained by multiple addressing of the primary in each instruction.

Case III.  $\underline{a} \gg \text{average } (b_1, \dots, b_n)$ . In order to achieve efficient operation we now require to carry out several of the  $b_n$  between each store access. This can only be done if a sufficiently large working space is provided in the zero-access store so that  $\sum \alpha b_n \sim \underline{a} \sim \sum \beta b_n$ . Such a system suffers from two difficulties. Firstly, if several  $b_n$  are to be carried out, the zero-access store must be considerable, and probably larger than sixteen registers in all. Secondly, the number of digits needed to specify the zero-access registers involved becomes considerable, and we must consider how to accommodate this. We have assumed that the primary is to contain both numbers and instructions, and have, so far, considered these on an equivalent (VonNeumann) basis. In order to continue to do so, it will be necessary to arrange that an 'instruction' word in the sense defined elsewhere above is of the same digit length as an integral multiple of number words. At each access to the primary store we now may obtain or dispose of either an 'instruction' or several numbers (or one very long number). Unless special precautions in the instruction code are taken, this method is both inconvenient and inflexible, and in any case will lead to equipment or time waste on transfer of single numbers. It is therefore, to be avoided. An alternative (Harvard) approach is to divide the primary into two sections with instructions in one and numbers in the other. For programs of varying length this is an uneconomical





use of the primary storage hardware, and furthermore, necessitates almost complete specialization of the decoding register in the zero-access store. With a limited zero-access store neither method appears very satisfactory. Clearly reference to more than one primary address in an instruction is of no assistance in this instance.

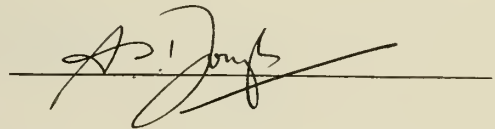
Let us now reconsider Case I for a system involving more than one reference to the primary store in each instruction. We may now specify operation of the computer with an instruction set up for decoding as:

Stage	(1)	(2)
1.	Select 1st primary store reference	Carry out the operations involving only zero-access store or transfer to 1st primary address involving the 1st primary store number etc. involving the $(p-1)^{th}$ primary store number
2.	Select 2nd primary store reference	
3.	etc.	
	Select next instruction	

We note that, if any primary store address,  $p_i$ , is associated with transference to the primary, then this must involve the contents of the zero-access register at the end of Stage  $p_i - 1$ , and the access time can only be used up efficiently if computation on the contents of zero-access registers can continue during Stage  $p_i$  independent of the transfer. As in Case III above there is a difficulty in matching the digit lengths of instructions and numbers, which is the more acute for large  $p$ , since the primary store is assumed larger than the zero-access store. The value of  $p$  is, of course, determined largely by the ratio of access time to Stage operation time, and we require that  $pa \sim \text{average } (\underline{b_1}, \dots, \underline{b_n})$ . The advantage of this system lies principally in the possibility of designing an instruction code which uses efficiently  $p$  references to the primary for numbers to a single reference for an instruction. The resulting saving is one access time in every  $p$  (which is not usually great in practice) access times, and the penalty to be paid is a certain redundancy or inflexibility in the instruction code. This system has, therefore, little, if any, advantage over a system involving only one reference to the primary per instruction, and the latter system has both simplicity of structure and flexibility to recommend it.



To sum up, the study so far carried out is of intentionally limited scope, applying only to a computer composed of (1) a limited size zero-access store, and (2) a primary store of isochronic access. Three cases can be distinguished, corresponding to rapid, equal, or slow access times to the primary storage relative to the speed of operation of the arithmetic, logic, and control circuits. It is concluded that in no case is any great advantage to be gained by referring to more than one address in the primary store in each 'instruction', provided some parallelism is permitted in the control. However, for cases in which the access time is of the same order as or greater than the time taken to operate on the contents of zero-access registers, the speed of computation may be increased by provision of an adequate zero-access 'working space', although this may lead to some inflexibility or complexity in the instruction code. It is, perhaps, worth noting that a computer having the latter structure can always be used as if it were single-address with a sacrifice of computing speed only, while enabling the maximum speed to be obtained where possible by skilful programming.

A handwritten signature in cursive script, appearing to read "A. J. Jones", is written over a horizontal line.









UNIVERSITY OF ILLINOIS-URBANA



3 0112 084228482