MASSACHUSETTS INSTITUTE OF TECHNOLOGY
PROJECT MAC

Computation Structures Group Memo 61

On the Number of Bits Required to Implement An Associative Memory

by

Robert M. Fano

August, 1971

<u>On the Number of Bits Required to Implement An Associative Memory</u>

by

Robert M. Fano

Let us consider an associative memory with a name space of size 2^p, a value space of size 2^n and a capacity of 2^m items (name-value pairs). The minimum number of bits, B, required to specify the contents of such a memory is the base-2 logarithm of the number of distinct mappings of 2^m (or less) points of the name space into the value space. The value of B can be readily computed to be

$$B = \log_2 \sum_{k=0}^{2^m} \frac{2^p!}{k!(2^p-k)!} 2^{kn}$$

and a close approximation to it is given by the upper bound (see 6.232 notes, Ch. 5, p.28)

$$B < 2^m(p-m+n)$$

This upper bound can be readily derived heuristically by noting that, if one were to store for each item the full name as well as the value, one would need p+n bits per item. However, the cells where items are stored have identifiable, distinct, physical locations (they are not subject to brownian motion) and therefore can be assigned order numbers from 0 to 2^m-1. Thus the location where an item is stored (its physical address) can be used to provide m bits of information about the item, and only p-m+n additional bits are needed to specify the item. This is indeed the case in a location addressed memory (p=m) in which only the n bits representing the value need to be stored.

The purpose of this memorandum is to describe a memory structure that requires (p-m+n+2) bits per item, and therefore comes within 2 bits per item of the theoretical limit.

<u>A memory structure</u>. An example of a memory structure employing (p-m+n+2) bits per item is illustrated in Figure 1. The 16 cells are numbered from 0 to 15 and i indicates the order number of a particular cell. The name of each item stored in the memory is divided in two parts: the prefix N_p consisting of the first m bits, and the suffix N_s consisting

of the remaining p-m bits. The bit F_i is set to 1 if there exist in the memory one or more item with prefix $N_p=i$; F_i is zero otherwise

$$F_i = \begin{cases} 1 & \text{if there exists } N_p=i \\ 0 & \text{otherwise} \end{cases}$$

i	F_i	T_i	suffix N_s p-m bits	value n-bits
0	1	1	2	
1	0	1	0	
2	1	0	3	
3	1	1	1	
4	0	0	3	
5	0	1	2	
6	0	1	0	
7	1	0	2	
8	1	1	0	
9	0	0	1	
10	1	0	3	
11	0	1	2	
12	1	0	4	
13	1	1	0	
14	1	1	1	
15	1	1	3	

Name of item N_p	N_s
0	2
2	0
2	3
3	1
3	3
7	2
8	0
8	2
10	0
10	1
10	3
12	2
12	4
13	0
14	1
15	3

Figure 1. Example of a memory with m=4

Items are stored sequentially in order of increasing name. Only the suffix of the name and the corresponding value are actually stored. Thus the suffixes corresponding to the same prefix appear in order in successive cells. The bit T_i is set to 1 in the first cell of a group of cells corresponding to names with the same prefix.

This memory structure has the following important property. The r^{th} "1" in the column T_i in Figure 1 marks the beginning of the r^{th} group of cells with the same prefix. The value of i corresponding to the r^{th} "1" in the column F_i is the r^{th} prefix (in increasing order) that exists in the memory. Thus this value of i is the prefix of the names in the r^{th} group of cells. For instance the 5^{th} "1" in column T_i in Figure 1 appears in the cell i=6. The 5^{th} "1" in column F_i appears in cell i=8. Thus the prefix of the name of the item stored in the cell i=6 is 8. The prefix of the name of the item stored in the next cell is also 8 because $t_7 = 0$. The arrows in Figure 1 indicate the prefix of the name associated with each cell.

Suppose a name with prefix N_p=j and suffix N_s=k is presented to the memory. The cell in which the corresponding item is stored can be found as follows:

<u>Step 1</u>: If F_j=0, no item with the given name is stored in the memory. Otherwise, go to <u>Step 2</u>.

<u>Step 2</u>: Find the smallest integer r such that

$$\sum_{i=0}^{r} T_i = \sum_{i=0}^{j} F_i$$

<u>Step 3</u>: Read the suffix N_s in the r^{th} cell. If N_s=k, return the value stored in the cell. Otherwise set r=r+1.

<u>Step 4</u>: If T_r=1, no item with the given name exists in the memory. Otherwise go to <u>Step 3</u>.