John Griffith

Stretch Memo #31

Suggested Memory Configuration

I. Rough Timing Principles

The basic aim of the memory design is to keep the arithmetic unit in full time operation. With present memory speeds, this is not possible, if one restricts oneself to strictly sequential decoding and arithmetic functions. Hence, the asynchronous non-sequential decoding technique must be used to overcome the shortcoming of long access times.

There are, in general, four distinct types of information to be stored in memory. These are:

(a) Operation codes

(b) Fixed addresses and parameters

(c) Variable addresses and parameters

(d) Data

Let us denote the required access times for these quantities by t_a, t_b, t_c, t_d respectively. Then, in order to determine what values these quantities must have, we must examine the times taken to perform arithmetic operations. We denote the time taken to perform a floating add by T_A, that for multiply by T_M, and that for division by T_D. Then, if for every number obtained from memory we perform f_A adds, f_M, f_D multiplies and divides, then the total time taken for arithmetic calculation, (denoted by T_c) is given by:

$$(1) \qquad T_c = f_A T_A + f_M T_M + f_D T_D$$

Note that when several pieces of data are processed together, the f's may not be integers. In any case, T_c really means the average calculation time per data fetch. Clearly, in order to keep the arithmetic unit in operation, it is necessary that:

(2) $$t_d \leq T_c$$

Let us now denote the actual access times for the various types of information by T_a, T_b, T_c, T_d respectively. These will in general be different from the required access times, and may also be different from each other, if a different kind of memory is used for each kind of information.

The difficulty in which we find ourselves at present is that:

(3) $$T_d > T_c$$

for the majority of problems of interest. However, for almost all of these problems, it is possible to obtain an effective access time t_d' such that:

(4) $$t_d' = t_d$$

i.e., the effective access time for data is equal to the necessary access time. Let the ratio λ_d be defined by:

(5) $$\lambda_d t_d' = T_d$$

Then λ_d may be thought of as a coefficient of simultaneity, which we hope to achieve by using the asynchronous non-sequential control system. (Definitions similar to these apply to cases (a), (b) and (c).)

Let us assume the data memory to be divided into N_d boxes to which it is possible to make simultaneous access. It is then possible to call from memory N_d pieces of data in the time T_d . Hence, for this case, the mean access time per data word is given by:

$$t_d' = \frac{1}{N_d} T_d \tag{6}$$

This means that a rate of flow of information from and to data storage of $\frac{1}{t_d'}$ wds/sec. may be achieved. If, however, a conflict occurs, so that the N_d words available each T_d are not the N_d required for the calculation in that time, then a repeated access must be made to one or more of the N_d boxes, and the calculation must wait for the accesses to be completed. Let the number of significant accesses made in time T_d be N_d', and let

$$\mu_d \equiv N_d / N_d' \tag{7}$$

be the <u>coefficient of degradation</u> of the simultaneity achievable under ideal conditions. The value of this coefficient may range from 1 to N_d , in order of increasing degradation. With this definition, the effective mean access time for a given data memory configuration (T_d, N_d) and a coefficient of degradation μ_d for a given problem, is:

$$t_d' = \frac{\mu_d}{N_d} T_d \tag{8}$$

Comparing (8) with (5), we observe that:

$$\lambda_d = \mu_d / N_d \tag{9}$$

The critical coefficient μ_d must be obtained by a detailed examination of the particular problem in question. It will naturally

vary from problem to problem on a given machine.

The asynchronous non-sequential decoding technique is used to obtain the benefits of simultaneous access to different memory boxes. This technique involves having the control unit look ahead of the point in the program at which the arithmetic unit is operating, in order to obtain data in advance. To a lesser degree, the control unit will look behind to store processed data. The amount of pre-and post-vision possible for the control unit depends on the number of words of data which it can hold at any time during a problem.

Let us assume that the control unit can hold R_d words of data, and that at a given point in the problem $R_d^{(i)}$ of these involve accesses to the i^{th} box of data memory. Then we have:

$$R_d = \sum_{i=1}^{N} R_d^{(i)} \tag{10}$$

Then if we denote $\left(R_d^{(i)}\right)_{max}$ by $\hat{R}_d^{(i)}$, the minimum time for data accesses for this section of the program is equal to:

$$T_{acc.} = \hat{R}_d^{(i)} T_d \tag{11}$$

Since, in this time, we have made R_d accesses, the mean access time per word for this set is:

$$t_d' = \left(\hat{R}_d^{(i)} / R_d\right) T_d \tag{12}$$

Note that the best mean effective access time occurs when all the $R_d^{(i)}$ are equal, and hence all equal to $N^{-1} R_d$ (by (10)). We see then, that if $R_d < N$, then some $R_d^{(i)}$ must vanish, and hence we do not have the optional use of an N-box stack of data memory. This fact sets a

feasibility restriction on the number of boxes with which a memory may be divided, insofar as the holding of a very large number of words of data by a control unit presents quite serious design problems.

All that has been outlined with regard to data applies with equal force to program information. Here, however, the essentially sequential nature of most of the control information enables one to assume with greater assurance that the various program memory boxes in a set will be called upon with approximately equal frequency, and that there will be little conflict. The worst cases will occur when the length of a loop is much less than the number of boxes of instruction storage.

Some general remarks may be in order about the various types of program information. The information which is most invariant in content and sequence is the operation code information. In addition, at least half of the address information is of a fixed variety, each unit of which may be associated permanently with an operation code, (e.g., many transfer addresses, origins of data blocks, etc.). These addresses should, from the point of view of coding convenience, be attached to their operation codes. The effect of this is that types (a) and (b) of information should be stored in the same memory, viz, instruction memory.

However, there are many variable addresses (mainly index register contents) to which access must be made in a highly repetitive and non-sequential manner. These addresses should properly be dissociated from operation codes and should be stored in a separate

memory, which we may call, modifier memory, or index memory.

II. Some Coding Examples and Their Memory Requirements

Let us consider a few simple examples:

(1) The inner product of two vectors; (2) Division of a row of a matrix by a fixed number; (3) Elimination of a matrix element by subtracting a multiple of one row from another row. In what follows, we assume each instruction to contain an operation, an address, and a tag.

(1) Inner Product

We wish to compute.

$$S = \sum_{i=1}^{N} A_i B_i \tag{13}$$

Let the locations of B_i begin at L°_B and be consecutive, and let those of A_i begin at L°_A and differ by M. Then the program would be roughly as follows:

LOC	OP	ADR	TAG
α	Load	L°_A	A modifier
$\alpha+1$	Multiply	L°_B	B modifier
$\alpha+2$	Add	partial S	
$\alpha+3$	Store	partial S	
$\alpha+4$	Test and Stepup	N	B modifier
$\alpha+5$	Stepups and transfer	α	A modifier
$\alpha+6$	Exit		

The references to various memories may be tabulated as follows:

LOC	Data	Instruction	Modifier
α	1 - A	1	1 - A
$\alpha+1$	1 - B	1	1 - B
$\alpha+2$		2	
$\alpha+3$		2	
$\alpha+4$		1	2 - B
$\alpha+5$		1	2 - A

The number of multiplies is equal to 1; of adds, also 1. There are 2 data references, 8 instruction references (S is assumed stored in instruction memory) and 6 modifier references. Hence:

$$(14)\quad f_A^{(d)} = \tfrac{1}{2}\ ,\ f_M^{(d)} = \tfrac{1}{2}\ ,\ f_D^{(d)} = 0$$

$$f_A^{(a)} = \tfrac{1}{8} \qquad f_M^{(a)} = \tfrac{1}{8} \qquad f_D^{(a)} = 0$$

$$f_A^{(c)} = \tfrac{1}{6} \qquad f_M^{(c)} = \tfrac{1}{6} \qquad f_D^{(c)} = 0$$

Let $T_A = .6$, $T_M = 1.2$, $T_D = 1.8$ (μsec.) and let $T_d = 8\,\mu$sec. In this problem, there need be no data access conflicts, so that $\hat{R}_d^{(i)} / R_d = 1/N$

Using these figures, we obtain:

$$(15)\quad T_C^{(d)} = \tfrac{1}{2} \times .6 + \tfrac{1}{2} \times 1.2 = .9$$

$$(16)\quad t_d' = \frac{1}{N} \times 8$$

Referring now to equ. (2), we see that

$$(17)\quad N \geq 8/.9 \sim 9$$

This shows that there must be at least 9 boxes of data storage if memory limitation is to be avoided.

Let us now assume that there are 7 instruction storage boxes so that each instruction of the loop of six may go into one box and the partial sum into the seventh box. This is the most favorable case. Then:

(18) $R_a^{(i)} = 1$, $i = 1, \cdots, 6$

$R_a^{(7)} = 2 = \hat{R}_a^{(7)}$

Assume: $T_a = 2$

Then:

(19) $t_a' = (2/7) \times 2 = .57 \mu sec.$

(20) $T_c^{(a)} = \frac{1}{8} \times .6 + \frac{1}{8} \times 1.2 = .22 \mu sec.$

Hence, we see that $2 \mu sec.$ memory is not adequate for instructions, even under the most favorable conditions. If we set $T_a = .5$ then t_a' becomes $.14$ and this satisfies condition (2).

Since only 2 modifiers are used in this problem, let us assume two modifier memory boxes with $T_c = .5$, to which we make 6 accesses. We have:

(21) $R_c^{(i)} = 3$, $i = 1, 2$

hence, $\hat{R}_c / R_c = \frac{1}{2}$ From this, we obtain:

(22) $t_c' = \frac{1}{2} \times .5 = .25 \mu sec.$

also,

(23) $T_c^{(c)} = \frac{1}{6} \times .6 + \frac{1}{6} \times 1.2 = .3$

Hence, .5 μsec. memory is just adequate for modifiers in this problem.

(2) Row Division

We wish to compute:

(24) $A'_{Kj} = A_{Kj} / A_{KM}$, $j = 1, \cdots, N$

A loop for this is:

LOC	OP	ADR	TAG
α	Load	A_{Kj}	A modifier
$\alpha+1$	Divide	A_{KM}	
$\alpha+2$	Store	A_{Kj}	A modifier
$\alpha+3$	Test and Skip	N	A modifier
$\alpha+4$	Transfer	α	
$\alpha+5$	Exit		

We assume A_{KM} to be in instruction memory. Then the access count is as follows:

Data	Instruction	Modifier
2	6	4

also:

(25) $f_D^{(d)} = \frac{1}{2}$, $f_D^{(a)} = \frac{1}{6}$, $f_D^{(c)} = \frac{1}{4}$

and all others are zero. Also, $\hat{R}_d^{(i)} = R_d / N$; hence, $t_d' = \frac{1}{N} T_d = \frac{1}{N} \times 8\,\mu\text{sec.}$ But $T_c^{(d)} = \frac{1}{2} \times 1.8 = .9$ hence N must again be about 8. A glance at the relative numbers of

instruction and modifier accesses shows that the same arrangements for these memories would suffice.

3. Row Elimination

We wish to compute:

$$A'_{Lj} = A_{Lj} - A_{LM} A_{Kj} \qquad j = 1, \cdots, N \tag{26}$$

A loop for this is:

LOC	OP	ADR	TAG
α	load	A_{Kj}	A modifier
$\alpha+1$	neg. mpy.	A_{LM}	
$\alpha+2$	add	A_{Lj}	A mod.
$\alpha+3$	store	A_{Lj}	A mod.
$\alpha+4$	Test and Skip	N	A mod.
$\alpha+5$	Transfer	α	
$\alpha+6$	Exit		

The access count is as follows:

Data	Instruction	Modifier
3	7	5

The f values are:

$$f_A^{(d)} = \tfrac{1}{3}, \quad f_M^{(d)} = \tfrac{1}{3}, \quad f_A^{(a)} = \tfrac{1}{7}, \quad f_M^{(a)} = \tfrac{1}{7} \tag{27}$$

$$f_A^{(c)} = \tfrac{1}{5}, \quad f_M^{(c)} = \tfrac{1}{5}$$

Hence, $T_C^{(d)} = \tfrac{1}{3} \times .6 + \tfrac{1}{3} \times 1.2 = .6$

We note that, in this problem, two out of every three accesses on record in the control unit must be to the same box. This means that: $\hat{R}_d^{(i)} = 2/3\,(R_d/N)$

Thus:

$$(28) \qquad t_d' = \frac{2}{3N} T_d = \frac{2}{3N} \times 8\ \mu\text{sec.}$$

In order for t_d' to be not more than $T_c^{(d)}$, we see that $N > 8$.

Again, the program and modifier situations are similar to those of the first problem.

We see then, that if one wishes to us an 8 μsec. access data memory, that at least 8 boxes would be necessary to keep the arithmetic unit busy while doing the quite important kinds of operations outlined above. This, however, necessitates keeping track in the control unit of at least 8 data references and/or words of data. Approximately the same number of instruction memory boxes of .5 μsec. memory are required.

These requirements imply severe complications in the control unit. Moreover, the circumstances outlined are the most favorable insofar as it was assumed that the mode of data storage permitted of conflict free access. In general, one may not expect

the minimum required data flow with 8 boxes of 8 μsec. memory. Calculations similar to those above would show that 4 boxes of 2 μsec. memory would be more than adequate. This is not to say that 8 boxes would be a superfluous or harmful division of memory, but simply that the control unit would be required to keep account of only four pieces of data.

Finally, if we assume that accesses must be made to data memory for input-output operations at a rate commensurate with that required for computation (in the examples given this is obviously so), then even 8 boxes of 8 μsec. memory would be completely inadequate. Even 4 boxes of 2 μsec. memory would just suffice.

John Gregory Postley