THE IMPACT OF CLASSIFICATION SCHEMES ON COMPUTER ARCHITECTURE

W. Handler

Rapporteur : Dr. D. M. Russell

1. Remarks on Classification Schemes and Formal Systems

Classification schemes, languages, and formal systems of all kinds have a considerable influence on our thinking. Structures which are inherently the subject-matter of a language as well as of classification schemes form the basic material of what can be expressed in a language or can be comprehended from its position in a classification scheme. The same statement seems to be valid for formal systems in a more specific sense. Thus the tool can be used in the application area for which it was created.

For example the Ricci-Calculus performs this role only in the area for which it was created, certain areas of physics and partial differential equations. Outside this area problems arise for which it is not suitable.

B. Whorf has said that language guides thought [11] and that therefore language sometimes prevents the appropriate solution of a problem being found. We must admit that in many cases a language (it can be referred to as a calculus or notation) can be a barrier rather than an aid in solving a problem. It is also true that a classification scheme can be a barrier, although it can provide an insight into the relationships between the elements of some group.

If such a classification scheme is to be applied to animals and plants, then the elements are existing objects and the scheme cannot completely fail, although the discovery of a new species can present difficulties in fitting it into an existing classification scheme. Such a scheme can be called a taxonomy, since all the species are considered to be descended from a single species, in accordance with the biological theory of evolution.

It seems more difficult to create a classification scheme, or even a taxonomy, for some area of contemporary technology. It is necessary to project future advances as well as placing existing examples in it.

The aim of this paper is to show that some existing schemes may fail to indicate the right direction for the development of computer architecture, as compared with a new and promising classification scheme introduced in [3], [4]. We would, however, not claim that the proposed classification scheme will cover all computer structures which will arise in the future. We do show that the proposed scheme does cover several very interesting structures which cannot be placed at an appropriate point in the schemes of Flynn [1] and Feng [2].

The justification of the proposed scheme is that it should

be useful in classifying structures and concepts which will emerge in the next years, and be of use to the designers of these structures. A further justification of the scheme is that the elements of the classification scheme can be decomposed by operations which are suitable for the purposes of the computer architect.

2. Contemporary Classification Schemes

Existing classification schemes differ in the information on which they are based. For instance M. Flynn [1] bases his scheme on a 'data stream' and an 'instruction stream'. By combining these simple concepts he can classify many of the new computer structures. In contrast, Feng [2] emphasises the number of bits which are processed simultaneously. These schemes are outlined in section 2.1 and 2.2 in order to contrast them with the scheme outlined in chapter 3. In section 2.3 the definitions of multiprocessing proposed by the American National Standards Institute [5] and by Enslow [6] are discussed.

2.1 Flynn's Classification

Flynn proposed in 1966 a classification based on the instruction streams and data streams. In the conventional Princeton type computer a single data stream is processed by a single instruction stream. This is described as SISD (single instruction single data).

In an array computer such as ILLIAC IV, a single instruction stream processes many data streams. Such a computer is known as SIMD (single instruction multiple data). In ILLIAC IV 64 copies of the same instruction are executed simultaneously by 64 arithmetic units. The Goodyear STARAN is also a SIMD computer. It differs from ILLIAC IV in many respects, in particular in being an associative array processor.

MISD is an abbreviation for multiple instruction single data. Some authors include various types of pipeline computers in this class though it is doubtful whether this is appropriate, and it is unsatisfactory because it does not distinguish between the three kinds of pipelining (see section 3.3 below).

MIMD is an abbreviation for multiple instruction multiple data. Here multiple processors are working on multiple data streams. The simplest case is where each processor is executing its own program on its own data. The processors can be connected via a bus system or can access multi-port memory. The classification does not contain any information about the type of connection used.

Flynn's classification is illustrated by Figure 1, where many contemporary computers can be classified by assigning them to one of the four vertices of a graph. However, the classification does not fully satisfy the needs of computer architects because it is not fine enough and because the interpretation of the class MISD is not clear (cf. [7]). In the literature many authors restrict themselves to the classes SISD, SIMD, and MIMD. A further

difficulty occurs if a computer contains both parallelism and pipelining.

2.2 Feng's classification

Feng [2] classifies according to the word-length, that is, the number of bits which are processed in parallel in a word, and the number of words which are processed in parallel. A computer structure is represented by a point in a plane (Figure 2) where the abscissa is the wordlength (normally 12, 16, 24, 32, 48, 60 or 64), and the ordinate is the number of words processed in parallel. The latter can be determined by the number of processors. For example C.mmp which contains 16 PDP-11's with wordlength 16 bits is represented by (16,16). The ordinate can also be determined by the number of arithmetic and logical units in an array processor. Thus ILLIAC IV is represented by (64,64).

Thus Feng's classification does not allow us to distinguish between multiprocessors like C.mmp and array processors. This caused Enslow [7] to represent C.mmp in "gang" mode by (16,256). But C.mmp in gang mode can be regarded as similar to ILLIAC which would give the point (16,16) which is the same as when gang mode is not used. The classification also does not distinguish between autonomous processors which execute programs and ALU's which execute operations, that is, it does not distinguish between processing levels.

The TIASC (Texas Instruments Advanced Scientific Computer is represented as (64,2048). The number 2048 is obtained from the 4 pipelines each consisting of 8 stages with 64 bits. However the number 2048 can be obtained in many ways, for example 8 pipelines, 8 stages, 32 bits. Thus the classification cannot represent a multiple pipeline structure like the TIASC accurately.

It is also not possible to represent the pipeline structure at the program level of PEPE. PEPE is characterised as (32,16), and the fact that each set of data (up to 288, each representing a flying object) is processed successively in three different ways is not represented. This is performed in three separate series of ALU's, and we can regard this as a three stage macropipeline (cf. section 3.3).

The lack of a rigorous definition of pipelining in the context of Feng's classification scheme leads to difficulties in classifying structures containing both pipelining and parallelism. Thus the scheme is not entirely satisfactory for the computer architect either.

2.3 Definition of Multiprocessing

Similarly to classification schemes, if definitions are too narrow, some viable computer structures may be excluded from consideration.

The American National Standards Institute [5] defines a multiprocessor as:

"A computer employing two or more processing units under

integrated control. " Manufacturers of systems containing two to four processors did not find themselves in conflict with this definition. The definition did not exclude future developments in computer architecture, but does not seem to have had any impact on contemporary architecture. Subsequently Enslow suggested a more detailed definition in his excellent book [6] which included

1. Two or more processors, having access to a common memory, whereby private memory is not excluded,
2. Shared I/O,
3. A single integrated operating system,
4. Hardware and software interactions at all levels,
5. The execution of a job must be possible on different processors,
6. Hardware interrupts.

We will concentrate on the first characteristic:

A common memory is mandatory. Such a structure is shown in Figure 3. It is easily seen that as the number of processors increases the congestion in the access to the common memory will also increase. Thus Enslow's definition seems to exclude system containing very large numbers of processors. Microprocessors costing a few dollars are now available, so that systems containing thousands of processors are now possible. Some of the more progressive projects of computer architecture such as PRIME [9] are also excluded. On the other hand some structures which satisfy Enslow's definition are subject to severe limitations on their expandibility and application due to their use of an expensive cross-bar switch [10].

Thus Enslow's definition also does not either satisfy the requirements of comtemporary computer architecture. We therefore propose that the definition of multiprocessing be extended to include systems containing two or more processors each of which has access to a subset of the memory blocks, and in which each processor can transmit information to any other via a chain of memory blocks. This definition would include the configuration shown in Figure 7.

2.4 The Influence of Classification Schemes and Definition

We have tried to show in the previous sections that definitions and classification schemes have their limitations and can prove a hindrance beyond a certain point. The computer architect should recognise when this point has been reached, and consider whether an entirely new classification scheme or definition is needed, which will ideally include all existing structures within a particular area and also all structures which will be considered in this area in the future. There is no doubt that one should consider very carefully the consequences of introducing a new classification, because of its possible educational and normative effects.

3. The Erlangen Classification Scheme

3.1 Introduction

The Erlangen classification scheme (ECS) was developed mainly in order to avoid the drawbacks of existing classification schemes, as outlined in section 2.

The basic requirements are

1. The objects to be classified should not be unnecessarily restricted. Any kind of computer system - in particular parallel processors, array processors, multiprocessors, pipeline processors must be classifiable in the scheme;

2. The classification must be sufficently fine to express those differences between the objects considered important;

3. The classification must be unambiguous.

The clasification scheme developed was also found to be a useful technique in computer architecture, in the sense that:

4. Composed computer configurations can be described by using operators which are applied to primitive elements of the scheme.

5. It can be used in evaluating architectural configurations, in particular with reference to cost.

6. It provides a measure for the flexibility of a system.

7. It provides a starting point for scheduling of flexible structures

The objects of the classification are not necessarily computers only. This will be amplified below. The flexibility mentioned in 6. above is connected with the fact that a computer can be represented by more than one point in the classification. The various points which represent a computer will be referred to as modes. The more modes a computer has, the more choice of mode it has for a particular application, and so the greater is its flexibility.

The classification scheme can be used for algorithms as well as for computers, and demonstrates the inherent partitioning of the algorithm into parallel sections and pipeline stages. The classification of algorithms must then be related to the classification of the computers on which they are to be run. In general, jobs must be investigated to identify the classes of the algorithms contained, and matched to the classes of the computers on which they are run. A more detailed discussion of this question will be given in another paper.

3.2 Parallelism

Our classification aims at characterising the parallelism and pipelining present in a computer system. The connections between the processors and the memory blocks are not included in the classification. It is assumed that the connections can carry the expected traffic and provide the required availability. In such a case the performance of the system is mainly determined by the processors, including their capability to transfer information

The classification is based on the distinction between three processing levels:

1. Program control unit - Using a program counter and some other registers, and, in most cases, a microprogram device, the PCU interprets a program instruction by instruction.

2. Arithmetic and logical unit - The ALU uses the output signals of a microprogram device to execute sequences of microinstructions according to the interpretation process performed by the PCU.

3. Elementary logic circuit - Each of the microoperations which make up the microoperation set initiates an elementary switching process. The logic circuits belonging to one bit position of all the microoperations are called an ELC.

A computer configuration can include a number of PCU's. Each PCU can control a number of ALU's all of which perform the same operation at any given time. Finally, each ALU contains a number of ELC's, each dedicated to one bit position. The number of ELC's is commonly known as the word-length.

If we disregard pipelining for the moment, the number of PCU's, ALU's per PCU, and ELC's per ALU form a triple, written

t (computer type) = (k, d, w).

We give some examples of the triple, where we assume that the reader is familiar with at least some of the computers:

t(MINIMA) = (1,1,1)

The "classical" serial computer. Some early European computers were of this form.

T(IBM701) = (1,1,36)

An example of the early "parallel" (on the 3rd level) Princeton computers.

T(SOLOMON) = (1,1024,1)

The historical concept of an array processor.

T(ILLIAC IV) = (1,64,64)

The famous array processor developed at the University of Illinois (without PDP 10).

T(STARAN) = (1,8192,1)

The well-known associative array processor (without host and sequential control processor) fully extended (32 frames of 256 bits each).

T(C.mmp) = (16,1,16)

The Carnegie-Mellon University multi-mini project using 16 PDP-11's.

T(PRIME) = (5,1,16)

The University of California, Berkeley, project in which time-sharing is replaced by multi-processing.

The different systems exhibit different kinds of parallelism, which is uniquely attached to one of the three levels. The numbers which make up the triple show this directly.

At first sight, the triples are able to classify all viable structures, particularly in regard to parallelism. But although parallelism is the most important phenomenon in contemporary computer architecture, pipelining must also be considered. The examples above exhibit parallelism but not pipelining. In the next section the classification is extended to include pipelining.

3.3 Pipelining

Pipelining can also be implemented at the three levels described in section 3.2, i. e. 1. PCU, 2. ALU, and 3. ELC.

For example level 3 pipelining is the well-known pipelining of the arithmetic unit used in the CD STAR-100 and the TIASC. The STAR-100 uses a four stage pipeline and the TIASC an eight stage pipeline.

An arithmetical pipeline can be regarded as a "vertical" replication of ELC's, compared with the "horizontal" replication used in a parallel ELC. It is therefore reasonable to multiply the number of ELC's, w, by the number of stages in the pipeline, w', to characterise the ALU. For the TIASC we have then

t(TIASC) = (1,4,64x8).

The multiplication sign will be used at all levels to separate the number representing the degree of parallelism from the number representing the number of stages in the pipeline.

The next higher level of pipelining is instruction pipelining. This involves the existence of a number of function units which can operate simultaneously to process a single instruction stream. It is based on the inspection of instructions prior to execution to identify those instructions which can be executed simultaneously without conflict. This is done by a scoreboard, in the terminology of Control Data. These instructions are executed as soon as a suitable function unit is free. This

technique is referred to as "instruction lookahead", "instruction pipelining", or "parallelism of function units".

A classical example of this kind is the CD 6600 computer. Disregarding for the moment the input-output section (that is, the peripheral processors), the internal structure of CD 6600 with 10 function units becomes:

t(CD 6600 central proc.) = (1,1x10,60).

The 10 units in this case are highly specialised (for example floating point multiplication, integer addition, incrementation, etc.) and therefore a gain of a factor of 10 cannot be achieved. The real factor depends on the special program actually running. An average of 2. 6 is a typical figure according to information available from Control Data. A combination of several function units of the same type seems to be quite reasonable regarding the better utilisation of equipment on the one hand and the now available large-scale integration technology on the other hand. These latter considerations nevertheless are not directly a subject of this paper.

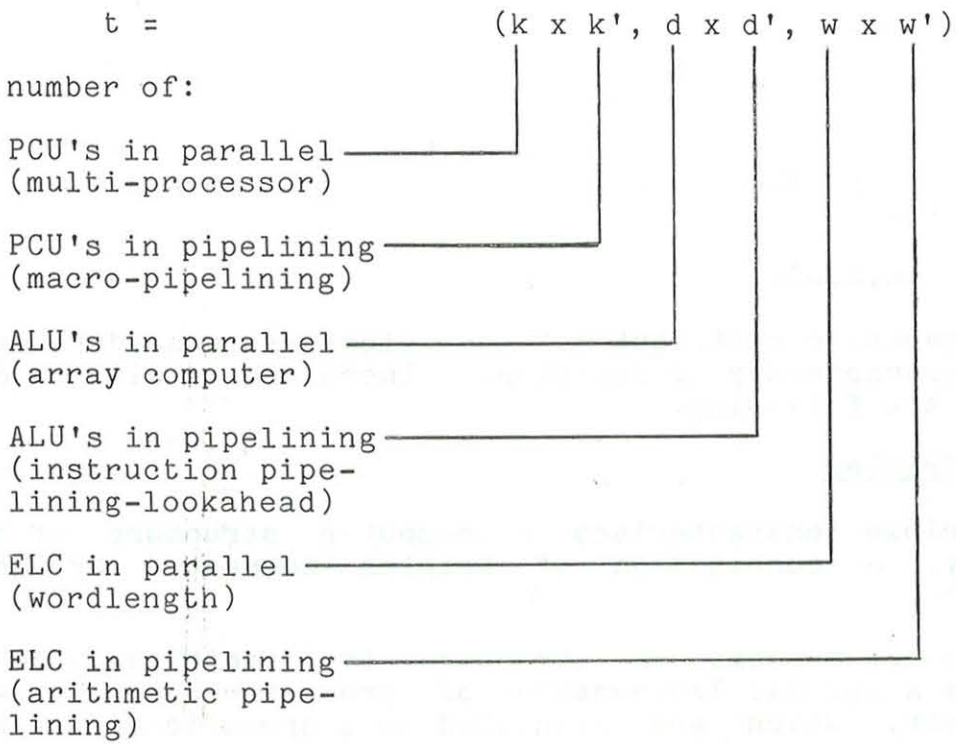
Finally, we have to consider the pipelining concept of level 1, which is so far not very known. This concept can be called "macro-pipelining" [12]. Assuming that a data set has to be processed by two different tasks sequentially, then it can be performed in two different processors, each one processing one task. The data stream then passes the first processor (1. task), is stored in a memory block, which the second processor also has access to, and will then pass the second processor (2. task). Since both processors can work at the same time (on different data), the effective processing speed can be in an ideal case doubled in comparison with the use of only one processor. In such a way stepping from processor to processor the data are 'refined' [12] by or are 'integrated' [13, 8] in the case of ordinary differential equations.

The PEPE array (without the host installation) then is characterised as

t(PEPE) = (1x3,288,32) (3-fold macropipelining).

Summarising, the triple has been extended to a sixtuple to incorporate pipelining. Nevertheless, we will continue to refer to it as a triple because the three levels of consideration (as introduced in 3. 2) suggest that we think in three terms, which have to be extended in some cases by an additional term, attached to the other value (of the same level) by using the sign x.

The triple now reads as follows:



All entities are independent of one another. All combinations therefore can appear.

Regarding the 'completeness' we claimed in section 3.1, we would have to prove that, apart from the three levels mentioned in section 3.2, no essential other level can be defined, and that there are also no phenomena apart from parallelism and pipelining. This is not done in detail here, because this paper centers on another topic, the impact of classification schemes on computer architecture. But there is some evidence regarding the completeness of our classification. While there are some modifications in details, according to how the level 2 pipelining is designed, there are no doubts about the other levels. With respect to parallelism and pipelining there is an exclusive duality as is known from other fields of science where parallelism and serialism also appear.

Regarding the triple notation, we introduced the following simplications:

k=1, or k'=1, or d=1 etc. mean, respectively, the simple cases, in which no parallelism or pipelining appear;

we write them

(1xk',dxd',wxw') = (xk',dxd,wxw')

(kx1,dxd',wxw') = (k,dxd',wxw')

(kxk',1xd',wxw') = (kxk',xd',wxw')

(kxk',dx1,wxw') = (kxk',d,wxw')

(kxk',dxd',1xw') = (kxk',dxd',xw')

(kxk',dxd',wx1) = (kxk',dxd',w)

If there is any form of pipelining then the character x is preserved in the corresponding level. In the case of no pipelining the triple degenerates to

t(MODEL) = (k,d,w).

This convention contributes to the clearness considerably as well as to the transparency of notation. Therefore we will use this convention in the following.

3.4 Operations on Triples

As a triple characterises a computer structure of a certain homogeneity, a combination of triples connected by an operator can denote

a) a more complex computer structure (as exhibited, for example, by a special I/O section of processors or by a special host, which are connected to a specific computer configuration);
b) a selection of operation modes of a structure, which can be used alternatively, fitting to different needs, according to the algorithmic nature of different applications.

It should be noted in connection with b) that for any application there can exist a number of algorithms, each one fitting a different computer structure. For example, one algorithm which is a solution to a given problem can be highly suited for execution on a conventional Princeton type computer, while another may be better suited for a parallel or pipelining computer.

The complete structure of the forementioned computer CD 6600 would be represented using a multiplication sign x, as:

t(CD 6600) = (10,1,12) x (1,x10,60).

The first term on the right hand side of "=" denotes the existence of ten processors of a simple structure with a wordlength of 12 bits. The second term is the characterisation of the nucleus of the CD 6600, as it was given earlier. The multiplication sign exhibits the fact that all algorithms'(programs) must be forwarded through the peripheral processors first, in order to be processed in the central processor (1,x10,60).

Another example of comtemporary computer architecture is PEPE (Parallel Element Processor Ensemble). Its host is one CD 7600 with the characeristic

t(CD 7600) = (15,1,12)x(1,x9,60),

PEPE then becomes

$$t(PEPE) = (15,1,12)x(1,x9,60)x(3,288,32)$$

where the last term (x3,288,32) corresponds to the actual PEPE structure, As, in this example, a certain flow of information penetrates the three structures, the sign x is used between the corresponding terms.

The structures characterised by the primitive terms in these examples are very different. Therefore a further condensation of the presentation is not suggested. A further decomposition can be indicated, for example by the use of other operators, for instance in the special case of a CD 7600 by

$$(15,1,12)x(1,x9,60) =$$

$$\underbrace{[(1,1,12) + (1,1,12) + \ldots + (1,1,12)]}_{15 \text{ times}} x(1,x9,60),$$

where $(n,d,w) = (1,d,w)+(1,d,w)+\ldots(1,d,w)$.

We note that the operators x and + again reflect parallelism and pipelining in a certain sense. The last example shows 15 equal processors allocated in parallel. A given job (or task) will be forwarded to the central processor. It may also be necessary to allocate processors serially, if there are different tasks to be performed one after another. This is supported by the use of functionally dedicated processors, specialised to the respective task.

The last operator we have proposed so far is the 'alternative' operator v, which is to be understood as an 'exclusive or'. For the c.mmp project which can be used in three different kinds of operation modes, an expression becomes:

$$t(c.mmp) = (16,1,16)v(x16,1,16)v(1,16,16).$$

Similarly, the EGPA project (4x4 array of processors, having a 32 bit word-length, described for example in [13]) reads

$$t(EGPA\ 4x4) = (16,1,32)v(x16,1,32)v$$
$$(1,16,32)v(1,512,1).$$

The last term of this expression denotes the operation mode "vertical processing" in which the 16 processors are used, each as if it consisted of 32 one-bit processors working in parallel. 16 processors then result in an ensemble consisting of 16x32 one-bit processors. Information then is oriented to one-bit vertical streams (items) and the machine-word of the memory becomes what is called a 'bit-slice' in associative processors.

The operator v visualises alternatives regarding the processing modes which can basically be used. An extended operator + can be used for a further partitioning of a system in which the

ensemble is working. Scheduling algorithms have to be developed which has to centre on the best utilisation of the system with respect to a given set of jobs. The scheduling problems, however, are not covered by this paper.

A remark on the 'flexibility' should be added. The number of available processing modes of a system seems to be a reasonable measure for its flexibility. Therefore we define (F=Flexibility):

$$F(t(MODEL)) =$$

$$\left|(k\ xk', d\ xd', w\ xw')\ v\ (k\ xk', d\ xd', w\ xw')v. \ . \ .\right|$$

where $\|\ \|$ gives the number of triples connected by the v sign.

For the examples presented above we have:

$$F(t(C.mmp)) = 3 \text{ and } F(t(EGPA\ 4x4)) = 4.$$

In this section we have tried to show that a classification scheme becomes operable if it is carefully chosen.

Nevertheless, it is not the aim of this paper to introduce ECS completely. We have used it as a further example of the discussion about the 'impact of classification schemes on computer architecture'.

4. Summary and Outlook

We have shown that ECS can be used for structures, which cannot be adequately represented by any of the systems mentioned earlier (chapter 2). Although we do not claim that ECS is the only possible classification scheme, we have found it useful for evaluating computer structures, throughput, flexibility etc.

In this respect ECS seems, as briefly presented here, to be an approach which can become a viable design tool. It classifies enough objects and it does not limit too seriously the set of objects. The only limitation we perceive so far is the inherently binary nature of the definition of w (wordlength). If a computer is based on another modulonumber system, then we would have to slightly modify the ECS as presented.

If, for historical reasons, we have to, for example, include the old mechanical calculating machines of Charles Babbage, then it would be necessary to extend ECS. Also excluded from ECS are computers of the analogue type. But this limitation seems to be quite natural in that analogue data processing is quite different.

The only criticism which at this time can be made within the aims of this paper could centre on the number of levels we introduced in chapter 3. There we defined a triple according to three processing levels. If perhaps in a later step of evolution a level above the program interpretation level will be created, then we would have to extend the triplet to a quadruplet.

But precisely this step to achieve a new level of computer structure is a real evolution step we are searching for at present. It was exactly for this that the classification scheme has been developed as a tool. About such an evolutionary step a decision cannot be made in advance. It is rather the ECS classification scheme and the operations defined on the elements (triples) which seem to be the appropriate starting point for investigations of that kind. We hope that ECS will not limit too narrowly a future development, for it includes all structures which so far have been proved to be viable examples of computer architecture.

Acknowledgement

The assitance of Mr. R. K. Bell and Dr. V. Sigmund in the preparation of this paper is gratefully acknowledged.

References

[1] M. Flyyn, "Very high computing systems", Proc. of the IEEE 54 (1966), 1901-1909.

[2] T. Feng, "Some characteristics of associative parallel processing", Proc. of the 1972 Sagamore Comp. Conf., Syracuse University, 1972, 5-16.

[3] W. Handler, "On classification schemes for computers in the post-von-Neumann era", in: D. Siefkes (ed.), GI-4. Jahrestagung, Berlin, Okt. 1974, Lecture Notes in Computer Science 24, Springer, Berlin (1975), 439-452.

[4] W. Handler, "Zur Genealogie, Struktur und Klassifizierung von Rechnern", Parallelismus in der Informatik, Arbeitsberichte des IMMD, Universitat Erlangen-Nurnberg, 9 (1976)/8, 1-30

[5] "Vocabulary for Information Processing", American National Standard, X.3.12-1970.

[6] Ph. E. Enslow, Multiprocessor and Parallel Processing, John Wiley and Sons, New York, 1974.

[7] P. H. Enslow, "Multiprocessors and other parallel systems - an introduction and overview", W. Handler (ed.), Computer Architecture, Workshop of the GI, Erlangen, May 1975, Springer Verlag Berlin (1976), 133-198.

[8] W. Handler, "Aspects of Parallelism in Computer Architecture", Proc. of the IMACS (AICA)-GI-Symposium on Parallel Computers, Parallel Mathematics, Munich (March 1977), North Holland Amsterdam, to appear.

[9] H. B. Baskin, Borgerson and Roberts, "PRIME - a modular architecture for terminal-oriented systems", SJCC 1972, pp. 431-437.

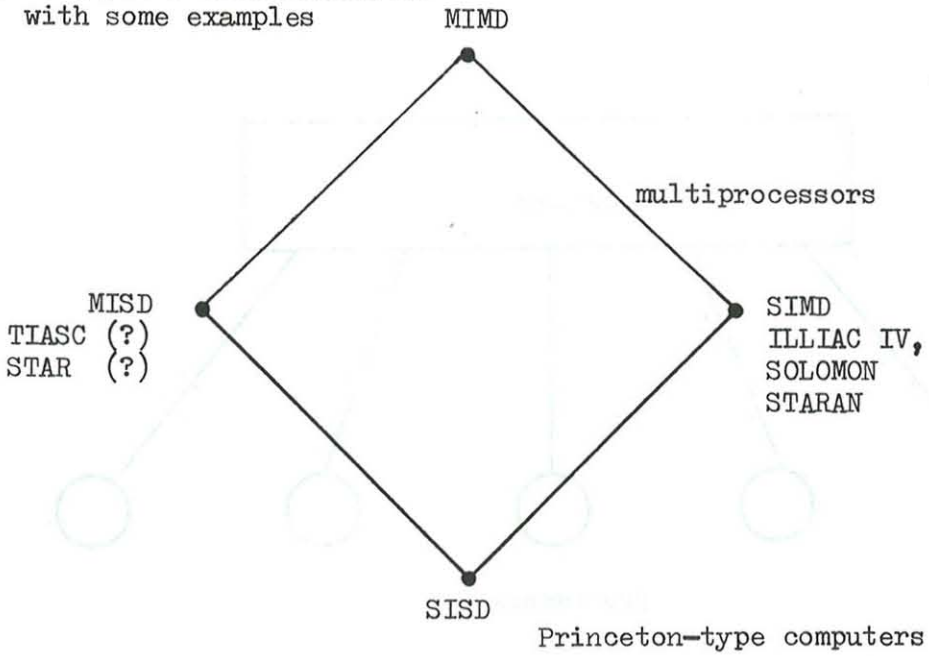
[10] W. A. Wulf, C. G. Bell, "C.mmp - A Multi-Mini-Processor", AFIPS Conf. Proc. FJCC 1972 41, pp. 765-777.

[11] L. Whorf, "Language, Thought and Reality", M. I. T. Press, Cambridge, Massachusetts, 1963.

[12] W. Handler, "The concept of macro-pipelining with high availability", Elektronische Rechenanlagen 15 (1973), pp. 269-274.

[13] W. Handler, F. Hofmann, H. J. Schneider, "A General Purpose Array with a Broad Spectrum of Applications", W. Handler (ed.), Computer Architecture, cf. [7].

Figure 1 : M. Flynn's classification
with some examples



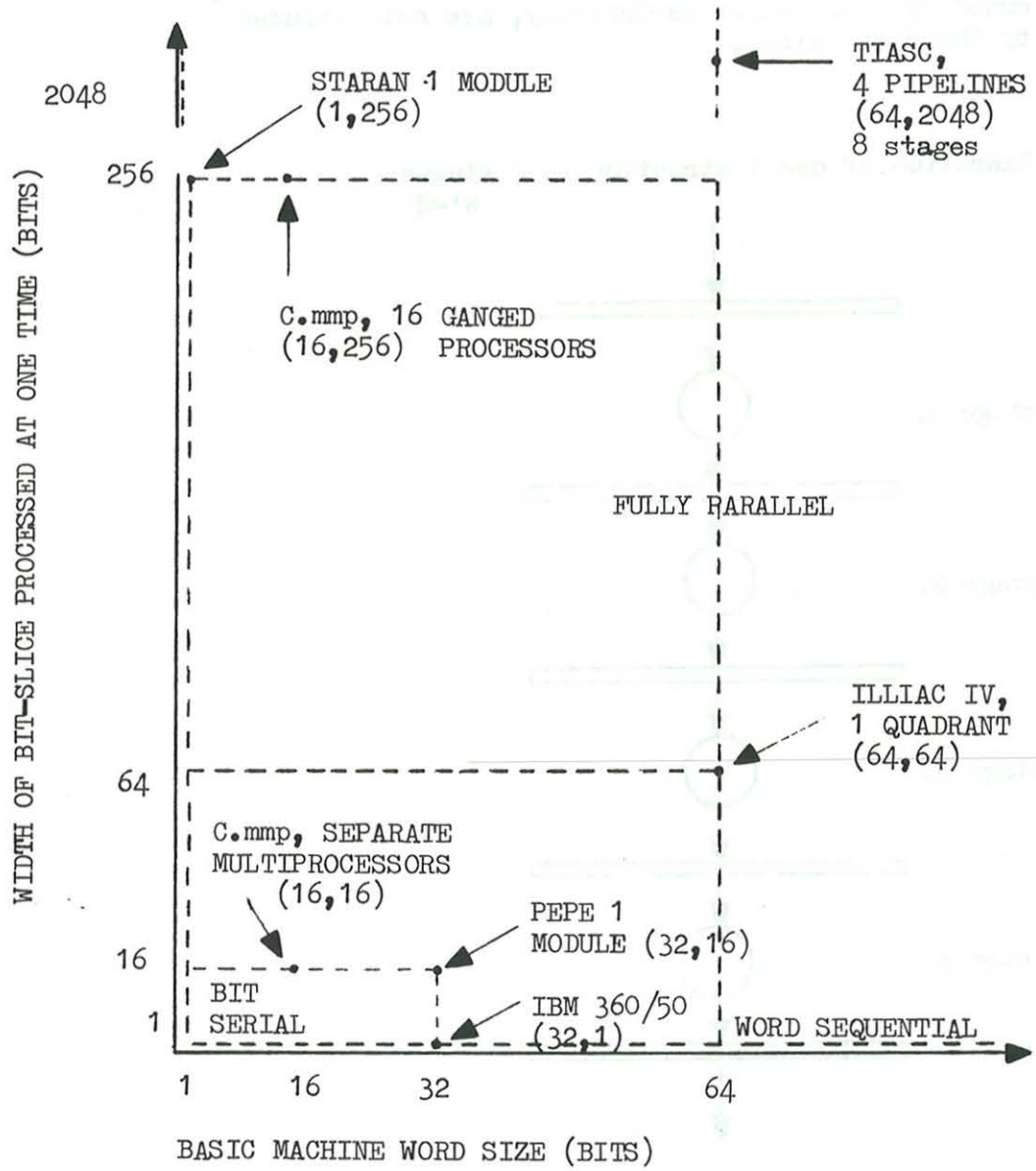


Figure 2 : T. Feng's classification
with some examples

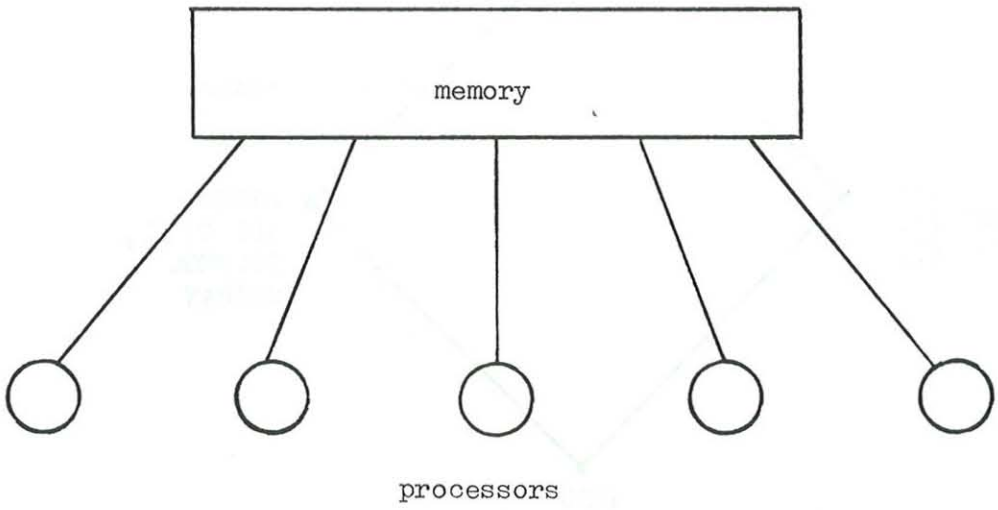


Figure 3 : P.E. Enslow's definition of a multiprocessor leads to "one common memory block" (private memory blocks, owned by a processor exclusively, are not excluded by the definition).

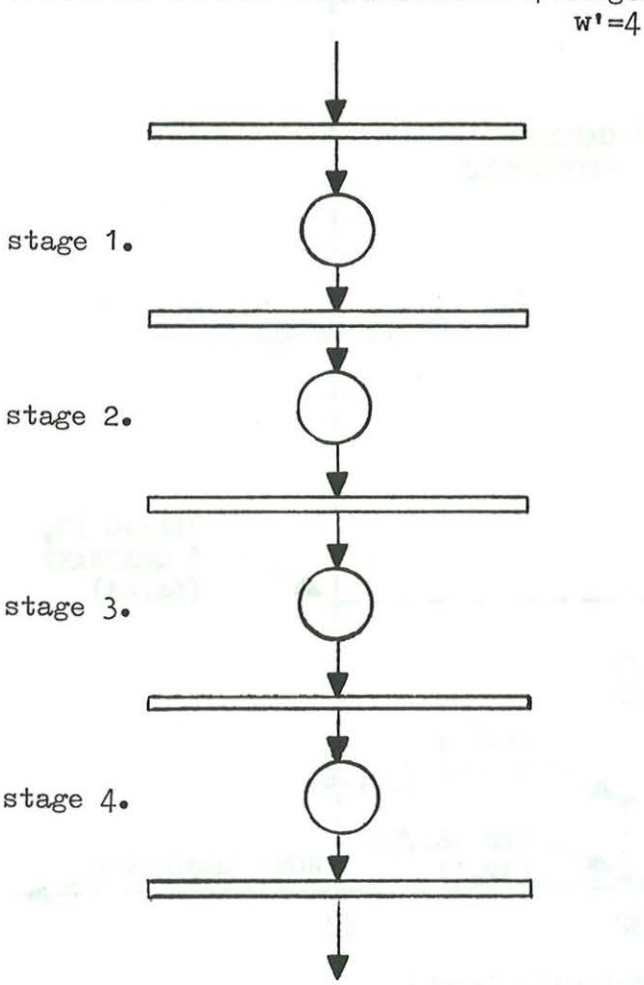
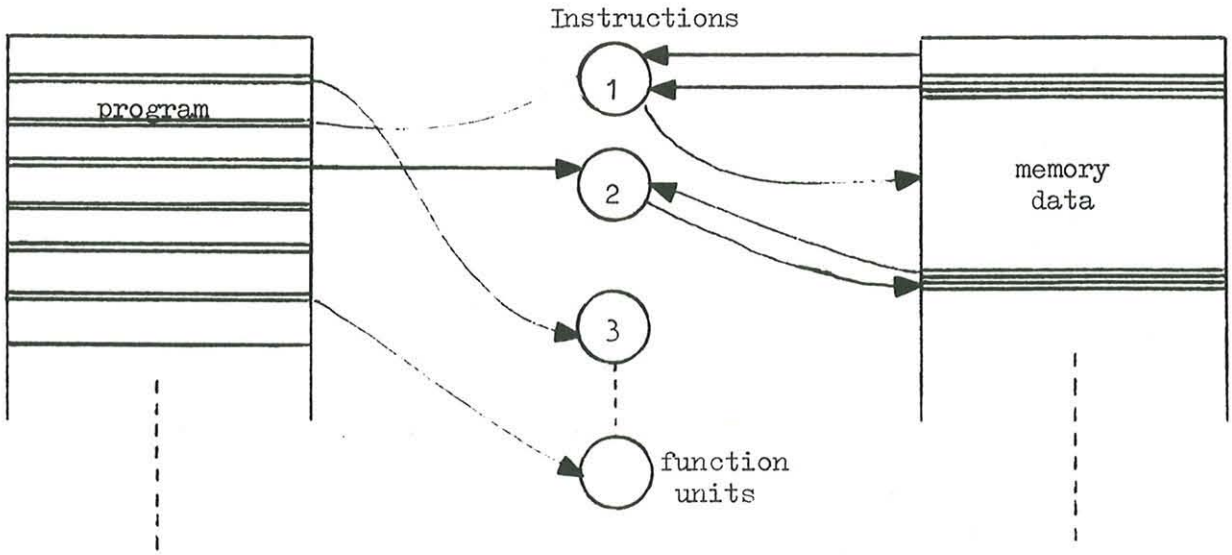


Figure 4 : Arithmetic pipelining (level 3 pipelining)

Figure 5 : Instruction-pipelining
(level 2 pipe-lining)



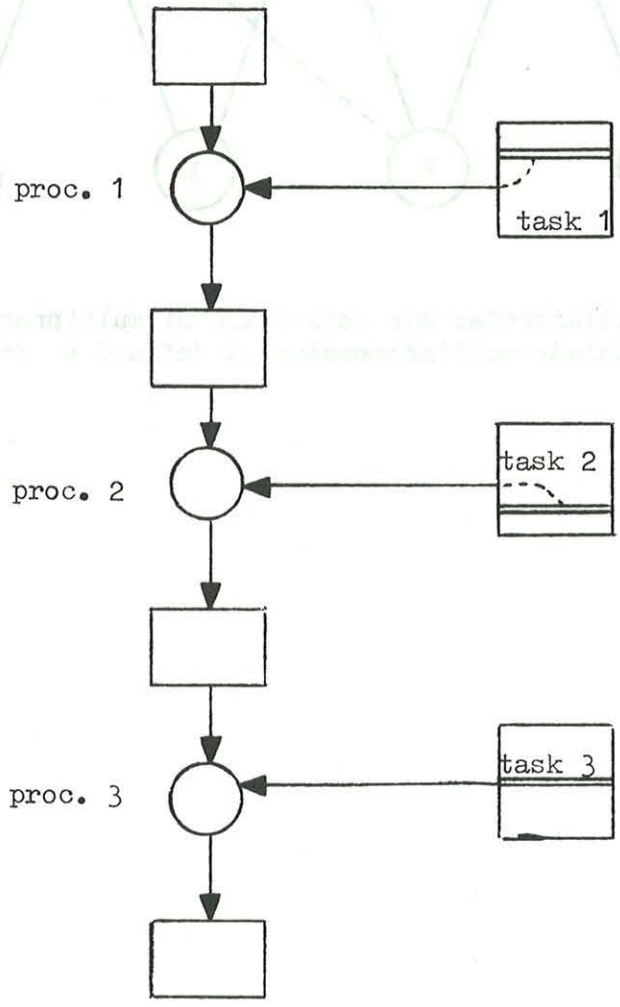


Figure 6 : Macropipelining (level 1 pipelining)

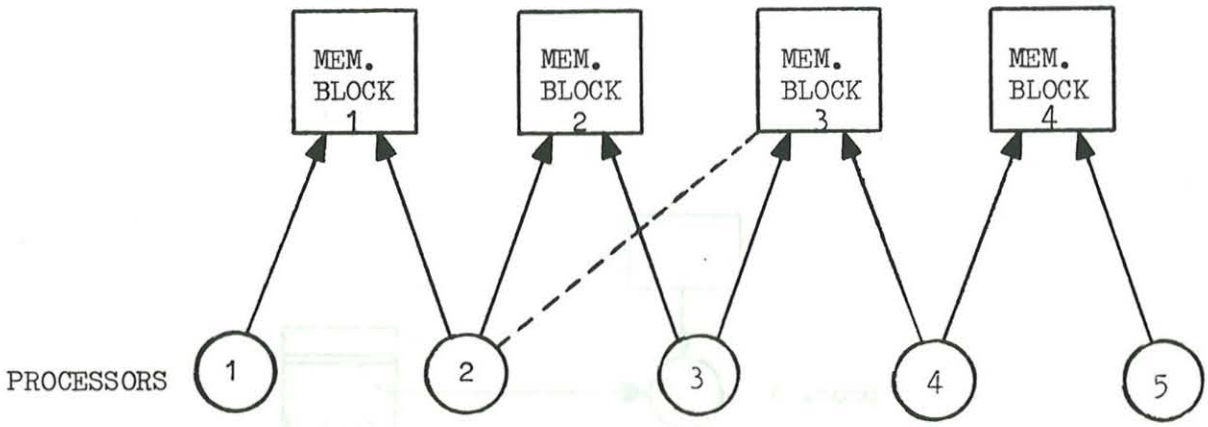


Figure 7 : This illustrates our definition of multiprocessing which does not exclude multiprocessing as defined by Enslow.