June 2, 1958

ANALYSIS IN DEPTH OF THE ITEMS LISTED IN PROPOSAL FOR SIMPLIFYING AND REDUCING TRANSISTOR COUNT OF SIGMA

1. Elimination of the Hamming checker, generator from the computer internal bus. (memory words, and exchange) The reasoning behind and implications of making this move are:

Little Need

a) The Hamming checking and correction system is being applied to an area where most generally we have experienced the greatest reliability in past machines. The most effective place for this type of checking would be the I/O.

Save Time

b) This is an expensive and time consuming operation. If it can be eliminated, not only will transistors be saved, but it may be possible to reduce the length of the time cycle in the I box thereby increasing performance.

Swap Time for Transistors

b1) An interesting thought eminating from item b is that it is conceivable if the I box cycle is reduced and performance increased, it may then be possible by removing one level (est.) of Look Ahead and saving more transistors, to bring the performance level back to where it presently is.

Parity should Suffice

c) A simplified parity checking system should suffice to replace the full Hamming checker based on the argument of item 1.

Only Unit with Error Correction

d) The Hamming checking system happens to be the only specific unit which presently has the ability to correct single errors. If this unit is removed, then another area must be found where single error correction facilities are available or else make a change in the wording or interpretation of the Sigma contract.

Hamming not Really Required

e) Errors that have been experienced in core memory systems once the system has been checked out and installed are not generally single bit errors. Line drivers, decoders, and sense amplifiers have caused more errors than the picking up or dropping of a single bit.

Challenge Reasoning Presented

f) Item e can be questioned on the grounds that we have no experience with the type of memory being built. There is however sufficient similarity so that intuitively the conclusion can be drawn that the relative parameters effecting reliability should not be substantially altered.

g) Elimination of the Hamming checker from the Sigma computer proper will immediately result in a transistor savings of about _____ units. If a substitution is made, relocating the Hamming or replacing it, then the transistors involved must be subtracted from the immediate savings.

Fundamental Transistor Savings

2. Elimination or making an optional feature of Decimal multiply, cumulative multiply and divide. The reasons behind and implications of making this change are:

a) The Sigma computer is now considered to be a product to satisfy the requirements of Scientific or Technical applications only. With this modification of competitive area, the existence of decimal arithmetic operations can be justly challenged. Binary VFL, and connective operations are still required, therefore providing a decimal add and subtract is of trivial cost and may possibly be used to advantage. The remaining decimal operations which can more readily be eliminated are multiply, cumulative multiply and divide.

Change in Computer Emphasis

b) The number of transistors that can be eliminated is approximately _____. This number is composed mainly of transistors that comprised the execution controls of the subject instructions. The basic decimal add facility is retained to provide simple add and subtract to aid in conversion problems.

Transistor Savings

c) Problems that could formerly be expediently done in decimal (normally short) may now be penalized in that it may become necessary to convert to binary, process, and perhaps reconvert to decimal.

Although this may be true, it is expected that this type of problem be encountered infrequently.

Alter Emphasis of Problem Attack

d) The inclusion of controls for decimal multiply, cumulative multiply and divide may possibly be made an optional feature. The suggested approach is to include the controls in the original design and layout, then not list the control components with the released system. The implication is that machines never desiring the eliminated facilities are less densely packaged or contain wasted space.

Optional Feature

3. Eliminate the ability to permit full word instructions to lie across word boundaries. Specify the restriction that full word instructions must lie

within word boundaries.

Reduce Complexity and Transistors

a) Applying the restriction will reduce the complexity and general confusion in attempting to understand the logic, switching and data paths of the Ibox. There will also result the elimination of approximately_____ transistors.

Assembly Program more Complex

b) Applying this restriction may make an autocoding program or assembly program a bit more complicated to write. There is nothing to indicate that this type of program would be impossible to write or when written would be substantially reduced in efficiency.

Main Program Inefficiency or Penalty

c) No Op. instructions would necessarily have to be inserted in a program at the point where the program breaks from a series of half word instructions to full word instructions if the last half word instruction ends in the middle of a word. The following two observations can be made: 1) Programs will not generally alternate between half word and full word instructions at frequent intervals. They will more likely execute loops or logic processing in one mode or the other and change mode only when it becomes necessary; 2) When a break is made from one mode to the other, the only concern is in the direction of converting from half to full word instructions. On the average 50% of these cases would require the inclusion of a half word no-op instruction - the case where the last half word instruction ends in the middle of a memory word.

Change in Emphasis

d) The emphasis of Scientific applications over commercial applications has resulted in suggesting the elimination of decimal multiply, cumulative multiply, and divide. A logical extension of this philosophy also implies less frequent use of full word instructions. Less frequent use of full word instructions also implies less frequent program changing from half to full word instructions thereby reducing the number of no-ops that would potentially be inserted into a program.

4. Eliminate the possibility that a 0.5 us memory should be part of standard installation, or standard optional equipment to the Sigma system.

a) Dr. H. G. Kolsky has shown that the inclusion of a 0.5 us memory does not necessarily imply an increase in performance, particularly when related to costs and possible substitutions for that cost.

Dubious Performance Increase

b) From information available the 0.5 us memories are not actually 0.5 us and are closer percentage wise to the 2.0 us memories than the given names imply. The capacities of the two memories are one to sixteen for a cost that is approximately equal. Therefore a summation reveals: costs about equal, capacity 1 to 16, speed of operation is not 4 to 1.

Cost Basis

c) The Sigma contract calls for inclusion of a 0.5 us memory and if it is eliminated, it is quite likely that renegotiation of the contract might be in order, or at least the substitution of some hardware or logic feature not presently listed in the contract to replace the 0.5 us memory.

Penalty of Elimination

d) Elimination of the necessity to provide facility for 0.5 us memory can directly result in the elimination of _____ transistors. The need for addition, etc. in instruction preparation, special bus switching, register entry etc. are eliminated.

Advantages

5. Eliminate the use of signs for index arithmetic, permit straight addition only with subtraction being done by addition of the complement of the number intended to be subtracted.

a) The acceptance of this restriction immediately results in a saving of approximately _____ transistors. Since only addition is permissible, no complementation is required at the inputs to the index adder, no recomplementation cycle has to be provided for, control logic is simplified, the carrying and manipulation of sign bits is eliminated and no test for a negative effective address need be provided for.

Transistors Saved

b) If this restriction is adopted, it may be more likely using emitter follower logic, etc. to reduce the time of index add cycle. This time reduction (and perhaps the time saved by the Hamming checker elimination) may provide the ability to reduce the overall time cycle of the I box thereby increasing performance.

Performing Advantage

c) Since only addition would be provided, the programmer cannot use a single numeric field to adjust direction of indexing. Two separate values must be provided, one to permit forward progress, the other to permit negative progress if it is necessary that this technique be employed. In certain instances a programmer can avoid the problem just described by using two index registers in place of one thus halving the number of independently available index registers, or maybe increasing the frequency with which available index registers must be set up for independent usage.

Programmer Penalty

d) No machine presently exists which provides the flexibilities and capacities of the Sigma computer. If a problem normally requires 4 index registers, eight Sigma registers can be used with seven remaining for multiprogramming, additional flourishes to the solution, etc. The net result can often be that no severe handicap is imposed upon the programmer and the problem may be run faster and at less cost depending upon increased I box performance and transistor count reduction.

Penalty Relationships

6. Eliminate Progressive indexing abilities and the ability to specify that a full word instruction may be immediate.

a) The adoption of this proposal will result in a transistor savings of approximately_____ units. Complexity of the I box will also be reduced in that it no longer becomes a problem to see what the 'P' field will turn out to be after indexing so that the proper handling and indexing of the instruction address can proceed.

Transistor Savings

b) Progressive indexing has always been subject to challenge primarily on the basis of doubtful advantage. Progressive indexing provides no unique ability, it makes possible a macro-ing of an index instruction with a process instruction, but provides no function that cannot otherwise be specified. Its existence is attributed to two things: (in this writer's opinion) 1. A compromise between two camps of opinion, with the resulting mental compromise that "If you accept this item, I will accept item x which you champion". 2. Demonstrating that this feature can be used to eliminate some instructions in certain areas of commercial applications.

Doubtful Advantage of Progressive Indexing

Emphasis Change

c) The recent change in emphasis; Sigma computer as a prime technical calculator, minimizing the commercial aspects, weakens the argument of item b) 2.

Doubtful Value of Immediates

d) The availability of an immediate modifier proves convenient when a constant to be derived from an address portion of an instruction as modified by an index quantity is desired. (either may be zero) How often is this desirable? Since an immediate modification cannot be logically defined when applied to a "to memory" type of instruction, its use is restricted to "to accumulator" type instructions.

Real Advantage

e) The existence of Progressive indexing and an immediate modifier can, at best, provide for the combination of certain types of instructions (index modification and processing) thereby reducing the total number of instructions comprising the program. The immediate modifier offers the advantage that for the particular case no data fetch need be made, nor is it necessary to provide a separate storage area for this particular class of constants.

Confusion Point

f) The P field or progressive indexing modifiers exist in the second half of a full word instruction. Both halves of a full word instruction are indexable. The following truths can be stated:

1) The exact nature of the instruction cannot be determined until the second half of a full word instruction has been indexed.
2) Logical rules of algebra cannot be applied to indexing the second half of a full word instruction since the indexed area is composed of independent fields through which carrys may propagate.
3) Extreme care will have to be exercised to prevent arriving at a P field result which is detrimental.

7. Eliminate or class as optional features the LZC (Left zero counter) and AOC (All ones counter) associated with the logical connectives operations.

Transistor Savings

a) The LZC and AOC are expensive units and if eliminated would result in a saving of approximately_____transistors.

Justification as an Optional Feature

b) The LZC and AOC are features which have never before been available to programmers, and although they are powerful tools, it is expected that programmers will not be making immediate effective use of these tools. It is the opinion of this writer that except for sophisticated computer users, these features will in larger percentage of cases be left idle while concerted efforts will be expended to work with the logical connectives. From this it is concluded that the LZC and AOC can be justly categorized as Optional Features. The intent is that the LZC and AOC be designed and layed out, but not included as part of the original model.

great balls of fire! this is the most — should be framed

no! not optional

Justification for Retention

c) The LZC and AOC provide information which cannot easily be derived by any other technique. The only justification for reducing the feature to an Optional level is its cost and lack of proof that the feature is a necessity. Its inclusion in the overall Sigma design along with the connectives raises the computer at least one level of sophistication above all existing computers.

you can't prove anything in a mathematical sense

// why not 2 levels

8. Eliminate or class an an optional feature the address comparison mechanism for multi program zone protection.

Transistor Savings

a) Removing the address comparison feature from the category of "integral part" of Sigma computer will result in a saving of approximately _____ transistors. *2700*

Question Multiprogramming

b) The address comparison mechanism is defined for program (memory zone) protection in machines that are being used for multi programming. The implication is that all Sigma computers are to be multi programmed. Is this a valid assumption? If one, two or several installations have problems that are sufficiently long running, to justify the elimination of the multi program concept should they be penalized with the cost of these extra transistors, this unnecessary programming obstacle, and perhaps increased execution time?

again just "perhaps"

most places will use this mode part of time, SHARE compatibility

Restricted Definition of Multiprogramming

c) An "inner group" multiprogramming concept can be defined where the memory protection can be made the responsibility of the auto assembler, supervisory program, or the automatic debugging program. This "inner group" concept permits multiprogramming only as it directly relates to the main program, i.e., setup routines, I/0 signals, exception case handling, etc.

More simply defined, an "inner group" multiprogram is defined as the sophisticated switching between the several sub routines directly related to the main line program.

Optional Feature

d) If the memory protection feature were made an optional feature, the feature would be designed, but the system would not include the hardware to perform the automatic protection unless requested by a particular customer(at an additional cost).

Compromise

e) A compromise may be a more acceptable solution. The compromise would include specifying a memory protection feature that is not so elaborate, effective, complex, or costly as the one presently proposed. The transistor saving would be reduced, but the overall effect would still be to reduce the total transistor count of the system.

9. Eliminate or classify as an optional feature the Elapsed time clock.

Transistor Saving

a) The elimination of the Elapsed time clock from the Sigma computer system would result in a saving of approximately _____transistors.

Challenge Validity

b) There is no known use being contemplated for the elapsed time clock in the areas for which the machines under contract are being built. There is no denying that potential uses can be listed by the dozen, but it is not presently known that the clock will be used.

Clock is Unique

c) The elapsed time clock does provide a unique tool for computer purchasers. It is the one feature that exclusively makes the machine fit into the category of real time processing. The effect of the clock cannot be duplicated in any manner - a clock must be provided or simulated for the situations where it is required.

Optional

d) Making the elapsed time clock seems to be a reasonable compromise. The customers not intending to use it will not have it, and those customers who desire it will find it available at a slight additional cost.

10. Restricting VFL operations to bytes of 4 and 8 only, and these bytes must lie on boundaries which are modulo 4 both in memory and in the accumulator.

Transistor Saving

a) Adoption of these restrictions can result in a transistor saving of approximately_____units.

b) If the above restriction is adopted, a define awkwardness will result in VFL programming. The awkwardness will result from trying to adapt existing control parameters (instruction addresses 18-19 bits, etc.) into fields whose length is modulo 4. Certainly some modification or re-definition of the machine will be in order, or at least the creation of some new instructions.

De-emphasize VFL Opns.

c) Single bit fields and non modulo 4 fields will have to be expanded to become modulo 4 by insertion of zero bits. The net result becomes a reduction in the density of information bits stored in memory. Complexities are also introduced in the I/O area where units functioning on a 5 or 6 bit character basis will have to be modified or converted by some intermediate buffer to be compatible with the processing capabilities of the computer.

Packing Density Reduced

d) The only byte size conversion that could be possible is from 4 to 8 and its inverse. Intermix of numeric and alphanumeric data for processing becomes less feasible.

Implied Restriction

e) A compromise that will permit more varied capabilities and, of course, reduce the transistor saving is to specify that the instructions mentioned apply only to the accumulator. Some true VFL attributes remain in retaining the ability to store and extract variable fields associated with memory, but these same fields in the accumulator must be modulo 4. Byte size conversion may be from anything (1-8) to 4 or 8 of the accumulator, and its inverse operation.

Compromise

f) If a decision is to be made on the basis of the original restrictions, it will probably be best to eliminate VFL from the computer completely and replace the void with some special instructions which will provide the missing abilities (i.e., Load Exponent, Store Address, etc.).

Personal Conclusion

11. Define, clarify and reduce the presently assumed level of completeness of checking.

a) The presently assumed level of checking is imposing a severe penalty in number of transistors required to reach the goals arbitrarily set. A large savings can be realized from ____ to ____ transistors if the level of checking can be reduced.

Transistor Saving

b) The present philosophy pertaining to the Sigma computer very simply states that the computer will be fully checked.

Review Philosophy

This philosophy can be expanded to have three parts. 1) full checking 2) error correction and 3) fault location. A clear guide has never been established to help the logical designer determine the type of checking system or the method for determining a reasonable cost for a checking system. The natural tendency has been to provide checking abilities definitely above a minimum level costwise, and with no method to judge adequacy of the applied checking system the whole system becomes shrouded in doubt.

Proposed Philosophy

c) To establish a criteria upon which logic designers can base their checking systems a Checking Philosophy is proposed. The net result of adopting this or any other set of rules should be a consistency of approach towards checking and a reduction in the number of transistors required for checking.

1. No proposed checking schemes shall include error correction except as it comes free from the checking system design.
2. No proposed checking scheme shall include fault location except as it comes free from the checking system designed.
3. No checking system shall contain more than 30% of the transistors comprising the area being checked.
4. The principle of "Borrow from Peter to pay Paul" may be applied to an area where a minimum checking scheme cannot be designed for 30% of the transistors of the area. If a minimum checking system costs more than 30% of the area being checked, the area may be extended to include another area in which the cost of checking was below 30%. Following this principle, the total cost of checking in the computer cannot exceed 30%.
5. No checking system shall be adopted if it severely increases the time of execution through the logic path being checked. Severe can imply 30% increase.

d) The following table may be interesting, it was originally prepared as an aid to the Analysis of checking problem:

	What kind of checking?	How much checking?	How to determine amount	Where should checking be done?	Test for adequacy of checking	Determining necessity for checking
Pure Checking	none parity casting out duplication etc.	none a little a bit more a lot	intuition discussion % of xistr logic % of over-all com - puter	I/O data paths mem & bus logic units controls	none intuitive no. of xistrs used program analysis statistical	none assume it is statistical reliability study
Error Correction	none special code try again - stop decode 2 out of 3	"	"	"	"	"
Fault location	none point to point decoding trap & analyze diagnostics	"	"	"	"	"

12. Eliminate the use of an adder for generating an interrupt address, substitute the prefixing technique.

Transistor Saving

a) Eliminating the use of an adder for generating an interrupt address will result in saving approximately _____ transistors.

Difference of methods

b) The addition makes possible the placing of the block of unique instructions related to the condition causing interrupt anywhere in existing memory. The prefixing technique restricts the positioning of this block of unique instructions to any module of 64 words. The worst conceivable loss a program can be responsible for is 63 unused memory positions. A program which cannot substantially reduce this loss is not well organized.

Faster Operation

c) A possibility worth mentioning is that the operation may be speeded up if no add cycle is required. The prefix technique calls for 12 bits of interrupt base address to be placed alongside the six bits from the Leftmost one identifier. It is not evident that this can have any effect on reducing the overall I time cycle.

13. Simplifying or restricting the existing definitions of instructions can result in a transistor savings.

Reduce No. of Formats

a) Eliminating some of the modifiers applicable to specific instructions, restricting the breadth of coverage of certain instructions and/or elimination of some of the desirable, but not really necessary attributes of certain instructions will permit an instruction class to fit into the format of other instruction classes. It is possible therefore to greatly reduce the number of different instruction formats resulting in the simplification of I box gating and control. Some ideas are available from proposals 0415 and 0417.

Transmit and Swap

b) The Transmit and Swap instructions are powerful tools for general programming and greatly facilitate such operations as set up and clearning for interrupt. There are times when a good thing can be carried too far. One possible case of carrying a good thing too far might be the ramifications specifield for the Transmit and Swap instructions. The following questions can legitimately be asked: 1. Is it necessary to permit this manipulation of half words as well as full words? 2. Does an immediate mode pay for itself? 3. Is the resultant aborted format just as easy to live with? (considerations to auto assembly and auto coding) Presently there are specified: Transmit; full, direct; half, immediate
Swap: full, direct; full, immediate

Branch Indicator or Bit

c) The most obvious deviant from a uniform format is the Branch on Indicator instruction. Another instruction is provided which serves exactly the same purpose - namely the Branch on Bit instruction. The largest difference between these two instructions is size - the Indicator is half word, the bit is full word. Two approaches can be suggested: 1. Eliminate the indicator instruction since the bit instruction with the numeric address of the indicator register will provide the exact same ability (at the cost of one additional half word). 2. Not all of the indicators are important enough to have an optional mask bit therefore why not lessen the density of bit packing in the indicator instruction by permitting reference only to the more important indicator bits thereby letting the instruction format become identical to other formats.

14. Eliminate Costly Instructions

Simplification of the instruction set by eliminating the more costly instructions can result in substantial transistor savings and in many cases do nothing in the way of reducing machine capabilities. (eliminate redundant or macro instructions)

a) The effective operand address is used to fetch from memory an instruction that will be modified and executed. The instruction counter is not stepped for this extra instruction fetch. Unique restrictions must be applied to the abilities of

of the instruction fetched for execution it may not: change the instruction counter contents, modify the indicator register, alter the boundary registers, effect the elapsed time clock, or effect the interruption address register. (Branch instructions cannot be executed since they by definition alter the contents of the instruction counter. It is also quite likely that the Look Ahead mechanism must be run out and maintained in this condition while an Execute instruction is being handled.

Execute Instruction

The Execute instruction has it's use in program tracing of other programs. It is therefore useful for program debugging and analyzing if a Master debugging or tracing program is available. What is the likelihood of having available a master debugging or tracing routine? What is the likelihood of individual debugging routines being written? Could not a reasonable degree of debugging be built into a Fortran type auto assembler without requiring the Execute instruction in the computer? No one can deny that the Executes are intellectually stimulating, but all can ask if they are useable.

Execute Uses

b) The effective operand address is used to fetch a word. The value field of this word is then used as an instruction address. The instruction is fetched and executed as in the Execute instruction. The value field of the word is stepped by one or two (half or full word instruction was fetched) and the word is returned to its original position in memory. The inclusion of the Indirect Execute instruction can nearly be defined as having one computer within another. The uses of Indirect Execute are the same as for the Execute instruction and the same questions can be asked regarding substantiation for inclusion.

Indirect Execute

c) The Store Address instruction provides that a portion of the value field of the index register specified by J replaces the address portion of the instruction located by the effective operand address. The portion of the value field that is transferred depends upon the instruction into which the number is placed.

Store Address

This is definitely a macro operation and eliminating this instruction does not effect the ability to accomplish the function, it will of course take more instruction space. Is this function used frequently enough to provide a hardware built in program? Could not the Store Value instructions be made to suffice? If some increased uniformity of formats be provided so that a single size address field can result, would not the Store Ad-

Questions

dress instruction and Store Value instruction be identical?

d) The Rename instruction has the following definition. The index word in the index register specified by J is stored in memory at the location defined by the refill field of the index word in index register Xo. The effective address (18 bits) then replaces the refill field of index word Xo and also specifies the place in memory where the new index word is obtained which replaces the index word originally in the index register specified by J.

Rename

The Rename instruction is then an automatic storing of, and reloading of an index word and its home location in memory. This is not really an instruction, this is hardware sub routine. Its usefulness is based on the premise of quickly and efficiently modifying and preserving an index quantity. The fallacy of this reasoning is that it is applicable to only one of the index registers. (It can be used concurrently for several index registers, but it would take a highly sophisticated programmer to keep things straight.) If the instruc- reverted to its original or earliest definition, then the ef - fect would be to halve the number of available index registers.

Since it is apparently more desirable not to halve the number of effective index registers, the Rename definition is now so restricted that its usefulness is minimized, perhaps not even existent.

e) The Load Indirect instruction specifies that an index word identified by J has its value field replaced by an instruction address which is located by the effective operand address of the original instruction. This is a macro instruction and reasonably acceptable, however there are some qualifiers to the existing definition which make one think twice. It is also specified that the instruction fetched to provide the ad - dress to replace the value field be inspected and decoded and if also a Load Indirect instruction the same initial operation is to be repeated. The implications of this definition are increased cost, increased complexity, and the possibility that the computer could run indefinitely and not produce useable results unless special precautions be taken. An interlock is further defined which will prevent the computer from running indefinitely, also an indicator position is exclusively devoted to this one instruction.

Load Indirect

1. This instruction is reputed to be a concession to Los Alamos, is it reasonable to penalize all future customers?

Instruction Set Reduction Simplification

2. Is there sufficient general purpose to justify the existence of this unique instruction?

15. Reductions can be made in the transistor count of the computer by eliminating from the instruction set those instructions which are little used, contain a degree of redundancy or provide a function than can readily be reproduced by a combination of other instructions.

Floating Point Single

a) A whole group of instructions can be eliminated on the grounds that their equivalent instruction in double precision, provides results from which can be extracted the single precision results desired. One point that should not be ignored is the desirability that the time required to produce the double precision result should not be significantly greater than the time to normally produce the single precision result. The instructions in this category are:

a1) F.P. Multiply
a2) F.P. Divide
a3) F.P. Add Magnitude
a4) F.P. Add

Load-Store Index Direct

b) The Load index and Store Index instructions are justified on the grounds that only a half word instruction is required to perform the function, this even in the light that two other instructions are available which accomplish the same thing. The first is the ability to Store giving the address of the index register. The second is either the Transmit or Swap ability. (The Swap provides more than just the Load or Store, it provides both.) The Transmit can even be faster and more powerful than the Load or Store index, i.e., Load 16 Ix registers. Using the Load Ix instruction technique; 16 instructions = 8 to 9 instruction fetches, and 16 Ix word fetches, total = 24 fetches. Using the Transmit technique; 1 instruction = 1 to 2 instruction fetches, 1 control wd fetch, and 16 Ix word fetches, total = 18 fetches. Net difference 6 fetches.

Symmetry Destroyed

An argument can be raised about the destruction of the symmetry of the instruction set if these instructions are eliminated. The fact is true, the symmetry will be destroyed, but the questions should be asked: Cannot this set be related to the Immediate Index instruction set which is symmetrical and yet does not contain either the Load or Store index? Should the set have been built up, so that it approached a symmetrical pattern? There are so many peculiarities in the system now, would it be burdensome to have a non symmetrical set of instructions?

Load-Store Refill Direct

c) The Load and Store Refill operations are convenient, but it appears that their justification for existence is a symmetrical instruction set and greater potential of use in a commercial application. (which aspect is presently deemphasized. Other instructions are available which can be utilized to provide this function. Among them are Load, Store and Connect.

Compare Counts

d) The Compare Count direct and Compare Count immediate provide no feature that cannot be reproduced by using other existing instructions. Its claim for existence is based on convenience, it is a macro operation whose use is in direct opposition to the philosophy of automatic detection of Count to zero. There is no substantiation on the basis of instruction set symmetry, in fact, its existence is counter to symmetry.

Compare Value Negative Immediate

e) The Compare Value Negative immediate instruction fits into all the classifications mentioned for the Compare Count, item d).

Interchange Augments

f) The Interchange Augment instructions are candidates for elimination because they provide no unique abilities and can not only be duplicated by other instructions, but the function itself can be eliminated in many cases just by judicious organization of the program instruction list. It is acknowledged that the cost of these instructions is small. Philosophically speaking, a lot of little raindrops can cause a flood.

Continued Compare

g) The Continued Compare instruction was defined to facilitate comparisons of fields whose precision is greater than one. The recommended procedure is to compare on the high order and follow with a continued compare of the lower order bits of a field. If the high order compared anything but equal, the Continued Compare became a no-op., otherwise it extended the compare operation. A serious complication arises since the sign of a field is appended to the low order bits, the initial compare cannot include sign relationships, consequently a false or incorrect comparison can result. The instruction Continued Compare as defined does not really provide any means for facilitating comparison of fields whose lengths exceed 64 bits.