MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Project MAC

Memorandum MAC-M-182

August 28, 1964

PRELIMINARY NOTES ON THE HARD MATTER FOR THE MAC 635

by

Edward L. Glaser

The following is a formal description of the hardware proposed for the augmenter portion of the MAC 635 system. It is beyond the scope of these notes to explicitly describe the use of each hardware function. The intent, in most cases, will be apparent to those familiar with modern software systems. These notes will still have to be enlarged before a complete record specification can be produced. However, they do form the foundation for such a specification. It is assumed that additional augmentation will already have taken place to the 635 system. These additions include such things as; 1. quick and total safe store of the processor state by means of a single instruction including packed index registers and the like, 2. 9-bit character mode as controlled by a single bit in the indirect word controlling the characters, 3. A 9-bit character option on the store character instruction.

In the following discussion certain oversimplyfing assumptions will be made originally to clarify the explanation. Where these assumptions are not truly valid but only an aid in explanation, they will be so noted. The current state bit in the machine used to determine master or slave, will have its meanings

redefined. Its true state will represent absolute addressing or relative addressing. This is similar to its present definition, however, the additional function of lock-out of certain commands in these relocatable or relative modes will no longer be part of its function. The master slave state of the machine is determined by another indicator to be discussed later. When in the absolute mode, it is assumed that the master state is forced. The master slave indicator is only effective in relative mode. In relative mode, relocation takes place in a way somewhat different from the current method of operation. In this mode two registers need to be spoken of, the first of which we will call the D register or descriptive register, the second we will call the P register or procedure register. (The procedure register does not exist in the form we will now describe, however it aids in explanation). At the beginning of instruction in the relative mode, bit 29, which is now an extension of the tag field of the instruction, is examined. If this bit is equal to zero, the operation proceeds in the following sense. It assumes that the full 18 bits of the address are to be used in a normal way and the "segment" to be addressed is that one in which the currently operating procedure is found. This particular mode of operation is also used with those instructions where the address portion does not refer to memory. (For a detailed description of segment see MAC memo TR 11 by Professor J.B. Dennis). If bit 29, however, is equal to one then the descriptor mechanism is enveloped. The six high order bits of the address are placed in a new 6-bit register we will call the effective name register or EN. The remaining 12 bits of the address are treated as though they were a full 18-bit address with a 6 high order position equal to zero. Address computation, that is, indexing if required, is applied. However, prior to the addressing of memory, the effective name register is used to address one of the 64 descriptors located in

table whose base is pointed to by the D register. The D register contains an absolute address which indicates the location of this descriptor table. The descriptor obtained from this table is what is called a segment descriptor and contains the origin of the segment desired. As a consequence, it supplies the base address for the relocation of this particular segment. (In point of fact, it points to a set of page descriptors. However, paging can be handled as a separate mechanism and will be discussed later). At present we will consider each segment to be in consecutive locations and will only be relocatable as a unit. If indirection is called for and the form of indirection is either I, IR, or RI, then bit 29 in the indirect word has the same function as in the case of the command word. Thus, in any of these cases, the entire 18-bit address can be used as specific address or a new name can be defined. As a fundamental rule of the system, an effective name remains in force until a new name is defined. At the beginning of each command execution the effective name register contains the name of the segment in which the program being executed is stored.

As was indicated parenthetically above, the segment descriptor does not point to the base of the segment but rather to a "page table." There exists a page table for each segment. The descriptor points to the base of this table and there is an entry in this table corresponding to each page in the segment. A page is defined as a set of words to be handled as a block. The function of the page table is to form a remapping index so that the various pages of the segment can be placed into any block of storage that is sufficient to handle it. The pages are located modulo 1024 in memory. The page index itself, may be either a part of the first page of the segment, a separate page belonging to that segment, or all page indexes may themselves

be pages attached to specific users or linked in other ways. The minimum size of a page index is 64 words and a page index may be located modulo 64 within a segment thus, 16 such indices may be carried within one page. The particular doctrine applied is left as a choice for the software systems designer. The specific entry of the index is defined by the 8 high order bits of the address. The entry found in this position of the page index supplies a new 8 bits which become concatenated to the ten low order bits of the effective address to produce the absolute address in memory. In review then, a 6-bit name, either the name of the procedure segment being executed or a name defined as a 6 high order bit of the address of instruction, are placed in the effective name register. This name picks a segment descriptor. The contents of the segment descriptor defines a base location for a page index as an absolute address. This portion of the address is concatenated with the high order portion of the effective address from the instruction to reference the page index. The entry obtained from the page index is now concatenated to form the high order portion of the final absolute address, the low order portion being defined by the original effective address. Thus, an access to memory, if descriptors are involved, requires two indirections even though no indirect tag was present in the instruction. To avoid this indirection, we employ a device called sticky registers (sometimes called a look-aside). These registers are effectively invisible to the programmer and their number is determined strictly by a question of economics and their efficacy in terms of program speedup. These registers act as a small associative store of a few words so that often used addresses are available directly in these registers. Each sticky register contains sufficient information to define the origin of a specific page in memory along with a segment identifier

referencing this page and other control bits. A simple wired-in algorithm is employed to determine which one of these registers shall be used if a new one is needed. Since descriptors are being used for accessing program as well as data, sticky registers can be used for accessing commands as well as data. As a consequence the P register is in point of fact a specific location in memory - PLOC - which contains the segment descriptor of the program currently being executed.

Both segment and page descriptors carry additional information, namely, the control bits. In addition, segment descriptors carry the information as to the number of pages that are present in this particular segment. This is used as a simple bound check to determine that an attempt is not made to address beyond the end of a segment. If this takes place, then a specific class of fault occurs which we shall call"within instruction interrupts or traps." This trap mechanism permits the interruption while part of the way through the execution of an instruction. It is applicable in this case and several others,as will become obvious. This specific mechanism,as to how this trapping is to be accomplished,will be discussed in a subsequent paragraph.

The page count information is not carried within page descriptors as each descriptor refers to only one page of a specific segment. The control bits however, are present in page descriptors as well as segment descriptors. The philosophy followed here can be described as the clearing of a bit to zero defining a restriction on the use of a particular segment or page. A page descriptor can add a restriction but it cannot take one away. Thus, if a particular bit is cleared to zero in the segment descriptor, then it is effective regardless of the presence or absence of this bit in the page descriptor. Thus, the effective control bits are formed by the oreing of

the compliments of respective bits of all descriptors, effective on a given access to memory. Nine control bits are apostulated, of which two are still defined as spares. The remaining seven are in order; write permit, slave access, procedure, master, table name, multi-page, and directed trap. It should be pointed out here that these are basic functions. It may be that some of the combinations that are possible here may not be found. However, at this point in time, it appears to be easier to define each function as a separate bit. The meaning of most of these bits may be fairly obvious but we will review them now in any event.

The write permit bit, if set to one, prohibits writing in this segment or page. The slave access bit set to zero defines that this particular segment or page can only be accessed when in master mode. (If this applies to a procedure descriptor then it means that we can only enter this segment while in master mode and any attempt to enter it from slave mode will force an interrupt equivalent to an MME. This will be discussed under interrupts).

Procedure defines that this is a procedure segment and therefore may be executed. Master if equal to one defines that this procedure segment or page is in the master state as opposed to the slave state. This is the additional indicator referred to earlier in this memo. The control bit called multi-page is only effective in a segment descriptor. This bit set to zero defines a single page segment and further, no page descriptor table exists for it. It is meaningless on a page descriptor. Likewise the control bit or table name is meaningful only in a segment descriptor. If this bit is equal to one the interpretation of this descriptor is changed radically. It is defined as pointing to a new table of 64 segment descriptors. The specific descriptor in this new table to be accessed is defined by the next 6 bits of the address prior to modification. If in turn, one of the entries in this table is defined as a table name descriptor, then the final 6 bits

of the address are used to access this descriptor and a new 18-bit address must be obtained from the program stream. This 18-bit address is defined as the high order half of the next word in the program stream. (As a consequence, table name descriptors can be used in depth to expand the equivalent addressing of the system to any desired length. As soon as one address has been exhausted another 18 bits of addressing is supplied from the program stream and this processing will continue until a descriptor is found that is a true segment descriptor, at which point the function reverts to a normal state. It may be that the new address of the instruction stream should be picked as soon as the table name descriptor is located in the first level. Further, it may be desirable to use the entire word from the program stream rather than the high order 18 bits. The method outlines here, however, is adequate and the other possible modifications are subject to economic and at timing considerations). The bit defining a directed trap, if equal to one forces a trap that is a within instruction interrupt. This trapping can occur either on a segment or page descriptor. If present, it terminates the operation at this point and forces the trap. Branching takes place to that location which is defined within the address portion of the descriptor. In the case of a segment descriptor, the branch must take place to a page contained within the segment. In the case of a page descriptor, the branch also must be within the segment, however, it may be possible to require that the branch even take place to a location within the page. The state of the machine, with respect to the special trap procedure not entered, is defined by the remaining control bits, that is, the master bit, write permit etc. If such a directed trap descriptor is encountered during the execution of a branch , which references what is thought to be a procedure descriptor,

the trap procedure is what is entered. This technique can be used as a generalized "presence" control guaranteeing that access is denied to specific pages or segments if these elements have not yet been brought into memory or if for some other reason, access must be denied.

Interrupts of the external type are answered in the present way, that is by a set of instructions placed in specific locations and are answered in absolute mode. Timer run-out, however, can interrupt a procedure while in the master state, although not while the machine is in the absolute state. A slight modification of the interrupt mechanism is necessary in order that the interrupt can be switched and so answered in the relative mode. In order to do this, the execution of single instruction should effectively force picking of two new interrupt instructions constituting a single interrupt cell from a location which is defined relative to the D register and a fixed distance from it. For example, location 64 through 127 for the external interrupts. Equivalently, master mode entry, while executed in slave, should force an entry into master mode, however, in relative location. A further execution of master mode entry while in master state but relative forces answering this in absolute state.

The handling of traps will be accomplished by means of storing the state of the control unit. The state is stored in words adjacent to the 2 word PLOC. A new PLOC is now defined by a pointer register with a new set of adjacent safe store locations. This pointer register is incremented once by each trap entered and decremented upon execution of "reestablish." Reestablish is a new instruction whose function is obvious. A minimum of sixteen such nested safe stores are feasible and automatic traps and a fixed absolute location takes place when the safe store stack is full.

The D register, that is, descriptor register, will appear to be much the same as the present relocation register. It will contain a base address which defines the base of the descriptor segment with the number of pages in

the descriptor segment. This descriptor segment, as well as containing the various segment descriptors and table descriptors, will also contain the descriptors relative interrupt cells, and safe store stack. It is sufficient for the two addresses in the D register to each have 8 bits long. The descriptor segment itself must have a descriptor if it is to contain any additional program or data or if it is to have manipulated being a program in relative mode. This descriptor is merely one of the 64 in the table. Loading and storing of the D register can only be accomplished in master mode either relative or absolute. This is equally true of storage of the pointer register which is part of the D register.

Finally, one additional change is necessary in the machine, namely, that instruction that stores the contents of the instruction counter plus two for subroutine entry and its corresponding instruction for return. The form should be modified so that the program name is stored along with the program counter. In the case of return, both the contents of the program counter and the PLOC must be restored.

To facilitate the manipulation of discriptors, each descriptor is actually a double word. The left or even full word, is an arbitrary symbolic name, while the right or odd name is the segment descriptor. In slave mode, all access to the odd words in the discriptor table is prohibited even if access to this table is granted by the descriptor. To facilitate the manipulation of descriptors, each descriptor is actually a double word. The left or even full word is an arbitrary symbolic name while the right or odd name is the segment descriptor. In slave mode all access to the odd words in the descriptor table is prohibited even if access to this table is granted by the descriptor for the discriptor segment. Reading and writing in the even locations, the symbolic names, is permitted. Writing in the even location

in slave mode automatically clears the next higher odd location to zero, thus forcing the master access' only condition on this discriptor and thus making feasible the changing of descriptor contents through symbolic name manipulation in slave mode. All locations beyond 127 in the first page of the discriptor segment are locked out from any access in slave mode. The remaining pages of the descriptor segment, if any, are under the normal control bit doctrine.

Care has been taken in this proposed system to keep the various elements of address manipulation and remapping separated so as to make it possible to perform future hardware modification, if this is deemed advisable. Further, it is hoped that there is sufficient flexibility in this design to permit wide latitude for future programming systems experimentation and implementation. Comment on this system is solocited and is necessary if we are to have an early effective MAC system running on the GE 635.