SPECULATIONS ON THE EVOLUTION OF AN ARCHITECTURE

A.G. Tagg.

Department of Mathematics, Statistics & Computing,
Oxford Polytechnic,
Headington,
Oxford OX3 OBP, U.K.

PRIME computers was formed in the early 1970s by a splinter group of hardware and software engineers from Honeywell. With them, they brought their ideas on minicomputers, based on their experience of Honeywell minis, and their experience of the MULTICS operating system.

It is possible, by examining the modes of operation of PRIME computers, to speculate on the development of the machine architecture. PRIME have developed four modes of operation, each one more powerful, in some respect, than the previous. Each mode of operation is called an "addressing mode", although the use of the term by PRIME implies far more than its conventional application to effective address formation. One interesting, possibly unique, feature of the PRIME's architecture, is that all four addressing modes are supported simultaneously on current machines, rather than each successive version of the architecture superceding the previous, as would normally be the case.

The modes, in order of appearance, are referred to as "S-mode", "R-mode", "V-mode" and "I-mode", the letters standing respectively for "sectored", "relative", "virtual" and "image". The first three modes follow the rationale of a one-accumulator-based architecture, but I-mode follows the ethos of current general-register-based machines. While S-mode was based solely on 16-bit instructions and integers, R-mode had some 32-bit instructions and real numbers. V-mode and I-mode are inherently 32-bit architectures with support for 16-bit working. There are no one-byte instructions and, apart from the shift instructions (and one character-handling instruction) immediate-mode addressing is not supported.

The first series of PRIME computers, was the '100 series, supporting S-mode initially, and then mainly R-mode (but see below). These machines were followed in the late 1970's by the '50 series, all inherently V-mode architectures. Developments over the last two or three years have shown a split between office machines (2250, 2550) and superminis (9650, 9750, 9950, 9955).

Reference to the literature indicates the use of the '100 series as a launching pad for each version of the architecture. The PRIME 300 was the R-mode prototype with software-controlled paging, while the 400 and 500 served as V-mode and I-mode prototypes respectively. Since the introduction of the '50 series, PRIME have made use of, what previously would have been considered, mainframe hardware techniques. All '50 models have an ECL cache memory (80 nanoseconds access) while the larger models are equipped with instruction pre-processing, error correcting MOS store (500 microseconds access) and a double-word (64-bit) storage fetch. Additionally extensive use is made of microcode ("firmware") notably for the operating system nucleus.

In the following sections, each of the four versions of the PRIME architecture is discussed in more detail. The discussion is meant to be indicative of the architectural developments and represents a review. For more detailed information, please refer to the references.

Sectored Mode (S-mode)

All PRIME machines are word-addressing machines, where the word is 16 bits. Two memory addresses are available with S-mode, 16K and 32K words; the addressing modes for these sizes are referred to as "16S" and "32S" respectively. Instructions are all 16 bits long and 16S addresses are 14 bits, 32S addresses 15 bits. The memory reference instruction consists of five fields: two bits indicate the use of indirect addressing and the use of the index register; another bit (the S-bit) is explained later. The remaining two fields are a 4-bit op-code and a 9-bit displacement.

The memory structure is similar to that of the PDP8, where memory is divided into 512-word "sectors" and a memory reference instruction addresses either sector 0 (denoted by S-bit =0) or the "current sector" (S-bit=1). In the first case, the displacement field gives the absolute address (where addresses 0-7 address the registers directly) and in the case of current sector addressing, the displacement is concatenated with the high-order bits of the program counter, to give the absolute storage address. Addressing outside the current sector is arranged by planting the address in Sector 0 and using an indirect reference through this Sector 0 address. Sector 0 is therefore used as a linkage, communication and global data area.

In addition to the memory reference instruction format, the repertoire includes "generic" instructions, (where the whole word is the op-code), shift instructions and raw input/output instructions. Amongst the generic instructions are the skip instructions typical of many minicomputers; in fact the conditional skips are the only conditional transfer-of-control instructions available under S-mode. An unconditional skip, an unconditional jump and a subroutine call ("jump and store program counter") complete the control transfer instructions. All out-of-sector references must be achieved through sector 0 address pointers.

It is interesting to note that all 16 bits of an address pointer are used, even when only 14 bits are needed for an address, as with 16S mode. In this case, the most significant bit signifies further indirection and the next bit, further indexing (performed before the indirection). Multiple-level indirection makes it possible to build horrendous linked address pointers, since there seems to be no limit on the level. Further indexing would make array access vectors possible, although these would necessarily be rather limited, since there is no facility for changing the contents of the index register half-way through an address calculation. Under 32S mode, the further indexing bit disappears and the address pointer comprises a further indirection bit and a 15-bit address.

It is very difficult to determine whether or not the early versions of PRIMOS were multiprogramming operating systems. The heavy use of absolute addressing makes protection very difficult and no context-switching instructions are evident. To contradict this evidence however, is the use of MULTICS as a base, with its sophisticated multiprogramming and timesharing facilities.

Relative mode (R-mode)

The two versions of R-mode are 32R and 64R for 32Kword and 64Kword memory sizes, respectively. It is possible to detect an "up-market" push already, partly from the increase in available memory sizes and partly from the extra addressing capabilities of the memory-reference instructions.

Addressing through the linkage sector (sector 0) is still available when the S-bit is 0. When the S-bit is 1 however, the displacement is no longer considered to be a 9-bit, unsigned value, relative to the start of the current sector ("sector relative") but instead, is interpreted as a signed 9-bit value, relative to the program counter, P (called "procedure relative"). Further, the whole range (-256 to +255) is not available, displacements in the range -256 to -241 are used to signal an "extended instruction".

With only a few bits to represent the op-code (4 in the case of PRIME) minicomputer manufacturers had to find some way of extending the instruction set, in order to provide the extra power that made their product more competitive. Whereas some manufacturers elected to make a single op-code denote an instruction group (e.g. the shift instruction group), with further detail being specified elsewhere, PRIME decided to move to two-word instructions. The two-word memory reference instruction holds the address, as a direct 15-bit or 16-bit value in the second word and hence frees the displacement field for use as an extension to the 4-bit op-code. This use of two-word instructions opens up the possibility of considerable hardware enhancements, even if it does take some of the edge off the increased memory size available under R-mode!

In 32R mode, an address pointer still needs only 15 bits for the actual address, and so multiple-level indirection is still available. In 64R mode though, all 16-bits are needed for the address and further indirection is not possible.

Other comments on S-mode and R-mode

Other hardware features are of interest here, since they further illustrate the standard philosophy of the minicomputer manufacture. On the PRIME 100 and 200 for example, integer multiply and divide hardware is available as an optional extra, while floating-point hardware was not available at all. These features were simulated by operating-system routines, rather than serviced by the hardware; this allowed the op-codes to be the same as for the larger machines, but an "unimplemented instruction" interrupt was raised, which caused control to be passed to a standard subroutine in the supervisor. Another aspect of the multiply and divide operations under both modes, is that they operate on a 31-bit double-length value, with the high-order bit of the second word being set to 0. (This representation, called by some a "32-bit value with a hole", is similar to that of the ICL 1900).

The addressing hardware calculates memory addresses modulo the memory size; in 32S and 32R modes for example, addresses are "truncated" (the manufacturer's term) to 15-bits. This produces a "wrap-around" effect that can have quite alarming results; in the absence of memory protection facilities, for example, it is possible to overwrite part of the linkage sector.

Other economies are evident from the absence of some instruction groups altogether; character-handling and packed-decimal facilities are two such, that you would not expect to find in the minicomputer repertoire. In addition to this, the registers occupy the first 32 locations of memory under R and S modes, and can be accessed as direct sector 0 addresses. This feature provides a cheap but viable alternative to a separate high-speed register file. Instruction execution times are limited by the speed of main memory, imposing a speed penalty that was deemed "acceptable" to most customers by many minicomputer manufacturers. (It is interesting to note that this feature was not adopted by the microprocessor manufacturers, where even the early versions had a separate register file.)

The PRIME 300

Although the PRIME 300 seems to have acted as an R-mode prototype, it had other features that distinguished it from the basic R-mode architecture, available by this time on the models 100 & 200. PRIME marketed a Virtual Memory (VM) feature and a Writable Control Store (WCS) option, that turned the 300 towards the architecture typical of many current super-minis.

The VM feature implemented virtual memory with paging under software control, and gave each user process a virtual addressing space of 262144 words. Although segmentation was not present, the page map allocated to each user process was also used to implement storage protection through a write enable/disable indicator. In addition, the manufacturer claimed that the use of a small (4-entry) content-addressable memory, reduced page-table look-up to only 3-4% of storage accesses.

The Writable Control Store option allowed user-written microprograms to be loaded into a random-access memory extension to the normal control store. (This feature is restricted to a few modern minicomputers only, another example being the Burroughs B1700 series and most customers have not found it necessary to exploit such flexibility.) The feature was enhanced by an Extended Control Store board that increased the size of this user-accessible control store further. The microinstruction width was 64 bits and the control store address space allowed for a maximum of 1024 such instructions.

A further feature of the Virtual Memory option was the emergence of the privileged operation concept, typical of many mainframe computers. This concept allowed in fact, three machine states: a privileged non-swapping state for the operating system kernel, a privileged swapped state for "outer" operating system functions and an unprivileged swapped state for user programs.

Other extensions to the PRIME 300 instruction set, were a set of conditional jump instructions and a set of stack manipulation instructions for implementing recursive procedure calls. The conditional jump instructions have passed into the standard PRIME repertoire, but have been re-named as "conditional branch"! The stack manipulation instructions are rather cumbersome to use and have been superceded by the inherent stack-based nature of the V-mode architecture.

Virtual mode (V-mode)

The arrival of the V-mode architecture (initially on the PRIME 400), brought the drastic change in PRIME machines, that heralded the '50 series. Only one version (64V mode) is available. In addition to hardware-implemented Virtual Memory, full storage protection is provided by the addition of segmentation. Under this philosophy, a program's code, its fixed (or "static") data and its working storage (or "dynamic data area") are kept in three logically separate areas, called segments. The hardware traps all attempts to address outside a segment (called an "address violation"). The additional feature, whereby the hardware traps attempts to store into the code (or "procedure") area or execute instructions from the data areas, has, rather surprisingly, not been implemented. (Programmers who have operated in multiprogramming environments where there is no form of protection, will be only too aware of the consequences of overrunning an array and corrupting their code - or worse, someone else's!)

Each of these three segments has its own base register; the code has the Procedure Base (PB - the program counter), the static data has the Link Base (LB) and the workspace has the Stack Base (SB). A further base register, available for programmer use, is the Temporary Base, XB. Switching from one user process to another, is called a "context switch", and is essentially achieved by saving the programmer's work registers and changing the contents of PB, LB and SB. A rather irritating feature of this context switch is that the hardware Procedure Call (PCL) instruction makes use of the two index registers, X & Y, and the temporary base, XB. This is tempered in part, by the addition of a local procedure call facility (called by the manufacturer the "short call") that does a branch and link on XB and leaves all other registers unchanged.

Additional protection is afforded by running the executive processes and the user processes in logically different storage areas, called rings. The executive runs in rings 0 & 1 and the users, in ring 3. The hardware allows processes to access code and data that are in the same, or higher-numbered rings and therefore, the executive can access a user processes segments, but not vice-versa. A V-mode address has up to four components; all have a 3-bit ring number, a 12-bit segment number and a 16-bit word number. Some addresses also have a 4-bit bit number, giving the net effect of making every bit in store addressable. Although the theoretical limit for virtual memory is 4096 x 65536 words (256M words), in practice a lower limit is set by PRIMOS; nevertheless each user process can have up to 16M words of virtual memory.

Three features of the segmentation and different addressing structure, merit further discussion; these are recursion, re-entrant code and fault handling. The two procedure linkage instructions, PCL & PRTN, provide a local, inter-procedure context switch, by changing the contents of PB, LB and SB from those of the calling routine, to those of the routine being called. The newly created stack frame, pointed to by SB, can be used in a recursive context, in other words a routine can call itself. In fact the use of these two instructions gives a naturally recursive routine linkage.

Since PB points to a read only code area, it follows that more than one process can use the same procedure segment, while at the same time having their own, different data areas. Thus a commonly used piece of code (the editor for example) can be held in one commonly-accessible segment, with each process accessing the code with its own setting of PB, while having private data areas. This feature is called "re-entrancy" and leads to a substantial saving in virtual memory space, since the shared (re-entrant) code can be held once only, instead of many times. In PRIMOS, the FORTRAN and COBOL library subroutines are held in shared segments.

In addition to the three or four components already mentioned, an address also has a "fault bit". When a routine calls an external procedure using a PCL instruction, the linker creates initially a null address pointer with the fault bit set. Many addresses can be resolved at link-time, in which case the appropriate entries are written into the address pointer and the fault bit is unset. If the segmented linker/loader (SEG) cannot resolve an address, the routine name is stored into the static data area and the fault bit is left set. At run-time, reference to any address with the fault bit set, automatically causes an interrupt to a standard pointer fault-handling routine. This routine compares the name that was stored by SEG, with a list of operating system routines and if one matches, the appropriate linkage to that routine is established. If no match is found, a standard pointer fault error is generated. PRIME have used this method to provide what is called "gated access" to operating system routines; it is seen as providing a more flexible interface than the former, more conventional, supervisor call instruction.

There are other differences between R-mode and V-mode. The instruction set is further extended, by limiting the displacement field of memory reference instructions to the range -224 to +255. In addition, the register file becomes a 32-bit structure external to main memory; a condition code and an extra index register (Y) are also added. There is a comprehensive set of "branch on condition" instructions that test a condition code or the contents of the short, long or floating-point accumulators. Other additional features include 32-bit arithmetic, using L (A&B concatenated) and a new 32-bit extension register, E, character handling and packed-decimal internal formats, together with appropriate instructions for their manipulation.

It is easy to see how the terms "supermini" and "midicomputer" have been invented to describe this type of machine. The features outlined above straddle the boundary between the minicomputers and mainframes of the late 1960's, blurring the differences that were so clear then and effectively rendering the use of the terms "minicomputer" and "mainframe" invalid.

Image mode (I-mode)

In their literature, PRIME make the purpose of introducing I-mode quite clear. It is to provide an alternative working environment to V-mode, where programs need to do a considerable amount of 32-bit computation. There is only one version available, called 32I mode and as with 64V mode, the numeric part of the name no longer refers to the memory size available, since this is not meaningful in a Virtual Memory environment.

The addressing structure, including the use of segmentation, is identical to that of V-mode, but startling changes to the register file and instruction set, make the I-mode machine look more like an IBM 370 than a V-mode PRIME. Under I-mode, the A,B,E,X & Y registers are replaced by eight general-purpose registers, named GR0-GR7. The term "general-purpose" alludes to the fact that all eight registers can be used as work registers or as accumulators; in addition, registers 1-7 can be used as index registers (note that register 0 is not available for this purpose).

The normal mode of working under I-mode, is with 32-bits; a 32-bit quantity becomes a "fullword" and the 16-bit quantity is referred to as a "halfword". Many instructions are available in both the fullword and the halfword forms. There are, for example, instructions to add to bits 1-32 or to bits 17-32 of a general register, designated "A" and "AH". Similarly "S" and "SH" denote full- and halfword subtraction, "C" and "CH" denote full- and halfword comparison, and so on. In line with these changes, a conditional branch instruction can test the contents of any general register against zero. With these examples, it is possible to gauge the power of the I-mode working environment.

All data types in I-mode remain the same as in V-mode, except that the designation of 16-bit and 32-bit integers change to "half-" and "fullword integers". The use of one level of indirection with or without indexing is still possible. The greater instruction repertoire is made possible by most instructions being 32 bits long and by using between 6 and 9 bits for the op-code. Since these instructions can designate any one of eight registers, three-bit fields are reserved in the instruction for this purpose. The I-mode machine is therefore a two-address machine, with instructions specifying both the source and the destination explicitly, as distinct from the one-address nature of the S- and R-mode machines, where the destination of many instructions (often the accumulator) was implicit.

Summary

One feature of PRIME's developments in architecture, which must be unique, is that, while the four architectures are so distinct that each represents a new generation of machines, compatibility remains. Under each development, with the exception of I-mode, we have seen how data structures and instructions under an earlier mode, have been included as sub-sets of the next. Indeed it is possible on the current '50 machines, to run programs under S,R,V or I-mode. In addition, where the addressing environment is the same, as it is with S & R and with V & I, it is possible to switch from one mode to the other at run-time!

So which is the true "native" environment? Well, on the smaller machines of the '50 range, 32-bit instructions have to be provided by executive subroutines. It is therefore more natural to think of these as V-mode 16-bit machines that have simulated I-mode features. On the larger machines, the norm is 32-bit working and all instructions are performed by hardware, so it is more natural to think of these as I-mode machines capable of emulating the other environments.

In summary, we must note that S-mode has, to all intents and purposes, disappeared. R-mode is currently being phased out in favour of V or I-mode. Perhaps these latter two architectures will co-exist for some time to come, but it is probable that the time will come when V-mode "meets its Armageddon" too. At that time, we shall be left with I-mode alone ....., or will we?

References

1. The System Architecture Reference Guide document PDR3060
   PRIME Inc 1983

2. The Assembly Language Programmers Guide document FDR 3059
   PRIME Inc. 1979

3. PRIME Concepts and Implementation Course notes
   PRIME Inc. 1983

AGT/February 1985