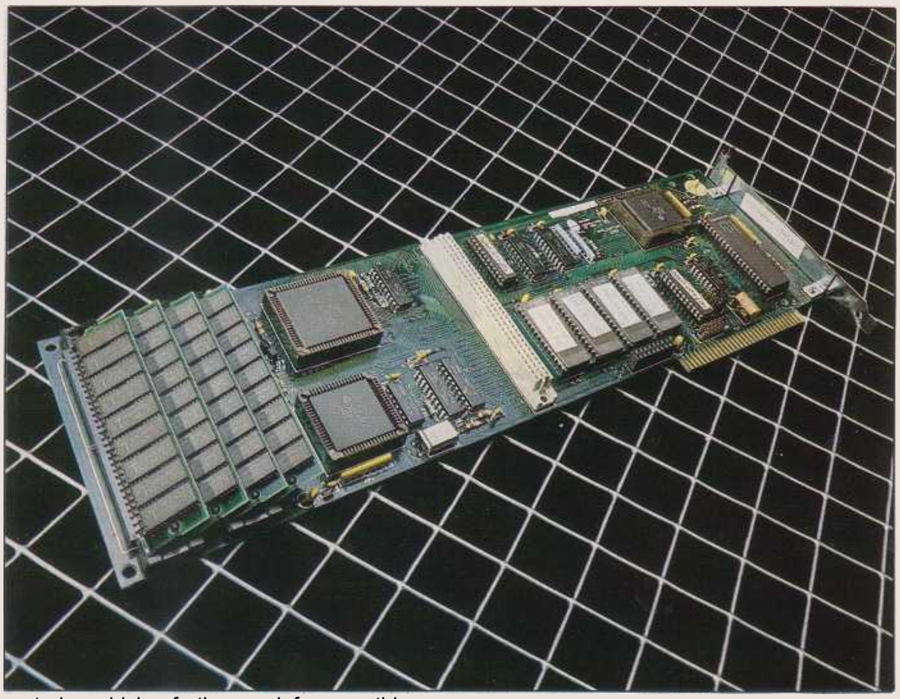
Acorn Springboard

Powered by Acorn's innovative ARM RISC processor, the Springboard coprocessor board turns your PC into a 'very powerful' workstation, albeit minus the stunning graphics. Such numbercrunching power doesn't come cheap, however, as Dick Pountain discovered.



The Acorn Springboard is a coprocessor board for IBM PC compatibles, powered by an Acorn ARM RISC processor. It contains either 1 or 4Mbytes of memory and communicates with the host PC through Acorn's 'Tube' parallel interface, originally developed on the BBC Micro. The Springboard is a selfcontained computer and can run programs quite independently of the host PC; this enables dual tasking, where the user runs an application on the PC while the Springboard runs its own program in the background. On the other hand, the Springboard Interface Software permits the board to swap data with the host PC using fast block transfers, to make DOS interrupt calls and use the host disks and filing system.

The 8MHz ARM processor is one of the fastest microprocessors available today, capable of sustaining 4MIPS (million instructions per second). Adding a Springboard to a PC or AT gives you a workstation with computing power, for suitable tasks, that rivals that of a VAX minicomputer. The sort of tasks that Acorn has in mind for the Springboard are mainly numbercrunching ones, especially for scientists and engineers; a Fortran compiler is available, as is a Lisp interpreter complete with the Reduce 4 Symbolic Maths system, which until now has been too much for a micro to handle. The Springboard contains a fast external bus connector which can take addon 'podules', and the first two podules to be released will be a very fast floatingpoint coprocessor and an IEEE parallel interface for data capture and instrument control, which further reinforces this orientation toward the laboratory.

One thing to be clear about right from the start is that the Springboard will not turn your PC into an Acorn Archimedes, for the simple reason that it lacks the Archimedes' graphics. The Springboard relies entirely on the host PC for screen I/O, and the Acorn VIDC chip which is responsible for the Archimedes' graphic displays is not even present on the board. Nor is it likely that graphics ability will be added as a podule, because the Archimedes attains its superb graphics performance thanks to the close coupling of its various components, and the podule interface is not fast enough to support the required level of VIDC/ MEMC communication. If you need the graphics, buy an Archimedes; Springboard is for CPU power only.

Hardware

The Springboard is a full length card which occupies a single PC slot. It contains a jumper block that allows alteration of the addresses of the I/O ports through which it will appear to the PC, a facility which I tested immediately because my external hard disk occupies the default port used by Springboard at 300hex. To inform the software of the new I/O address it is merely necessary to alter the contents of a text file, ARMREG.DAT, from 300 to 308, or whatever. This faecility allows up to four Springboards to be fitted in a single PC, all of which could run different programs simultaneously.

The board I tested was the 4Mbytes version, and its memory comes in the increasingly familiar form of four SIMM modules filled with nine 1Megabit chips each (parity checking is included, hence the extra one), and mounted in angled plastic grooves to reduce the board area consumed. The rest of the neatly laid out board contains the ARM itself, the Acorn MEMC memory controller chip which translates all the addresses fed to the ARM, and the Acorn IOC chip which controls the podule interface, a Tube chip and four hefty EPROMs containing interface software. Rather surprisingly, the EPROM does not contain Basic V, which comes separately on disk and is RAMresident.

Across the very middle of the card runs a 96pin female connector which is the podule bus. The podules will be piggybacked boards which occupy a second PC slot adjacent to the Springboard, both for reasons of mechanical rigidity and to draw their power directly from the PC bus. I was not supplied with any podules for review, but I do have the specifications of the floating point and IEEE podules.

The floating point unit eschews the better known coprocessors like the Motorola 68881, Intel 80387 or Weitek in favour of AT&T's WE32206 Math Acceleration Unit. This part was designed to complement AT&T's 32bit Unix engine, the 32200, and looks like a very serious device indeed. Fabricated in 1 micron CMOS, it performs single (32bit), double (64bit) and doubleextended (80bit) maths to ANSI/IEEE standard with onchip support for trig functions and square root. It uses a 32bit bidirectional data bus to communicate with its ARM host, and can be clocked at up to 24MHz.

The IEEE488 interface podule from Intelligent Interfaces Ltd is developed from one it produces for the BBC Micro, and can control up to 14 IEEE test and measurement devices, plotter or printers. It can transfer data at 250k per second and may be programmed in highlevel languages such as Basic, Pascal, C and Fortran in addition to Assembler.

Software

The Springboard interface software has been designed to be as nearly as possible transparent to the PC user, and it succeeds to a large extent. You can edit files, and compile and run programs on the ARM in an environment which looks very much like PCDOS v 2.0, rather than less familiar Acorn operating systems.

Part of the interface software lives in ROM on the Springboard itself. The rest takes the form of a hefty TSR (terminateandstayresident) program which runs on the PC host, called ARMINIT.EXE and a frontend command interpreter called ARM.EXE. This takes up 131k of memory and so it is not worth using Springboard with much less than a full 640k PC if you plan to run large PC applications too. ARMINIT loads itself and resets the ARM, reporting on its success as it does so; then it goes to sleep until summoned. The only small problem I found here is that the ARM will only initialise properly if ARMINIT is run from the directory in which the ARM files are stored (probably ARMREG.DAT is the important one); it doesn't like PATH specifications and hangs the PC if invoked from another directory.

To wake up the Springboard you type ARM at the PCDOS prompt, whereupon the screen clears and the prompt changes from your normal prompt to A*>. You are now talking to the Springboard's ARM rather than the PC's 8088. However, if you type DIR, a directory listing appears as usual, and what's more it is the listing of the directory you were in when you typed ARM. The file system has 'crossed over' from the PC to the ARM address space. This means for example that you could use a PCbased text editor to prepare program code, then type ARM to perform the switch and compile the resulting file with the ARM C Compiler, without any copying or conversion of files at all. The frontend program maintains its own default drive and directory which may be different from that set in DOS.

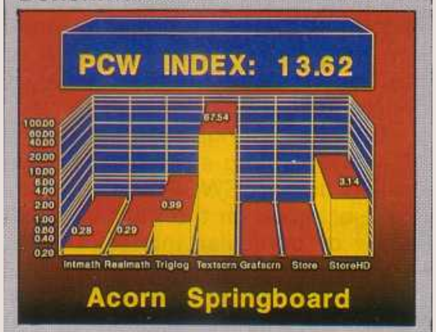
When you type EXIT to leave the Springboard, you are returned to DOS exactly as you left it. I mean *exactly,* for the interface software actually buffers the original screen and restores it so that you see the word ARM you typed to leave it. When you type ARM again the A*> screen is similarly restored exactly as when you left it. A seemingly trivial detail like this has a powerful effect on the ease of use of the software, because it creates the concrete illusion of switching between two real 'places', with which you can become familiar, rather than between two abstract 'modes'; it is precisely the same illusion that makes SideKick and other popup software so effective. Which reminds me: to my great surprise and pleasure, I was able to pop up SideKick while running the ARM without any trouble; this is quite a testament to the ' hygiene' of the interrupt handling in ARM. EXE.

I said that the Springboard screen would be exactly as you left it, but in fact there may have been some activity while you were away, because any program running on the ARM continues to run when you return to DOS. When you have launched an ARM program from the A*> prompt, you can exit back to DOS without waiting for it to terminate by pressing the CtrlEsc keys, leaving the program running. The converse is not true; a DOS program cannot be running when you activate the ARM because you need to be at the DOS prompt to do it, and nor does CtrlEsc interrupt DOS programs. It is possible to write DOS programs that can communicate with Springboard programs, but more of that later. The point is that the Springboard cannot be invoked from inside standard DOS applications. I had a problem with the CtrlEsc command, in that it wouldn't work while I had Borland's SuperKey loaded (in whatever order). However, I've had trouble with several other applications and SuperKey, which appears to be rather badlybehaved, and so don't feel inclined to put much blame on Acorn.

The ARM command can also take command line parameters which make it even easier to set off a Springboard program. Typing ARM E MYPROG at DOS causes an ARM program called MYPROG to be run on the Springboard but then exits immediately back to DOS; it's like launching a background task. ARM X MYPROG means exit back to DOS only when the ARM application MYPROG terminates. When an ARM application is running 'in background' it cannot do any I/O at all, either to screen or disk. I/O calls are suspended until you switch back to the ARM side of the fence, hence any application which needs to do screen or disk I/O is better launched with ARM X. However, a pure numbercrunching program can happily churn along in background with ARM E and will only print its result when you switch back. ARM R resets the ARM, the ARM I initialises the interface software without resetting the ARM and can be used to run multiple Springboards in one PC (by altering the port address in ARMREG.DAT).

The frontend program ARM.EXE which produces the A*> prompt supports 38 commands in all, and they are by no means all exact copies of their DOS equivalents,

Benchmarks



See 'Time trials', PCW, November, page 138 some being lifted straight from Acorn's ADFS. However, the main housekeeping ones DIR, CHDIR, MKDIR, RMDIR, and RENAME are the same and a DOS user will feel at home very quickly. Some of the commands perform operations which have no equivalent in DOS. For instance, CACHE <filename> <runname> loads an application into high memory and leaves it resident, after which it can be invoked without disk aceess by typing <runname>. With 4Mbytes of memory you can cache both the C compiler and the TWIN editor (and more besides) which speeds up the development cycle splendidly. Some of the other A*> commands are to do with debugging, and under DOS would be part of the DEBUG program.

To prevent confusion between executable files for the 8088 and those for the ARM, the file extension .AXE (by analogy with .EXE) is used for most ARM programs; there is no equivalent of a .COM file for the ARM. There is, though, a seeond kind of executable file called .AXH which contains a header that specifies its load and execute addresses, whereas .AXE files always load and execute at 8000hex.

Programmers' interface

Acorn has devoted as much effort to smoothing the program interface between Springboard and the PC as it has to the user interface. Programs written for the Springboard alone using C, Fortran or Basic V can use the PC host's disks and screen via the normal commands, just as if the PC were a terminal. All the PCW Benchmarks run exactly as published.

Writing programs which communicate between the two processors cannot be quite so transparent for fundamental reasons; when working with a coprocessor it is impossible to disguise the fact that there are two separate memory spaces, and that the operations of two independent CPUs need to be synchronised. The solution chosen by Acorn is effected through BIOS software interrupts, which is a very good solution given that this is a level at which IBM PC programmers commonly work.

The new PC BIOS interrupt 64H has 14 functions, which are called by the usual PCDOS convention of putting the function number in the 8088 AX register and then issuing the interrupt. Incidentally, this interrupt-driven interface means that it is not necessary (though it may sometimes be preferable) to write Assembler programs to drive the Springboard. Many modern PC languages, including Turbo Pascal, C and Modula 2, have the ability to issue software in-terrupts without machine code.

Several of the new funetions are to do with initialising and resetting the ARM, connecting the ARM to the host keyboard and screen, and passing commands to the executive firmware. For example, ARMCCE (function 4) connects the keyboard, inserts a command string into the keyboard buffer, and exits when the ARM program has read the buffer. The similar ARMCCAE (function 7) does the same but only exits when the ARM program terminates. ARMSERV (function 8) checks to see if the ARM program has requested a DOS service, and tries to satisfy the request. An ARMSERV request can fail because a DOS call is already being served or because the Tube is currently being read from. ARMPOLL and ARMEXEC (functions 9 and 10) are to do with running programs on the ARM; ARMPOLL checks whether the Springboard is waiting to do I/O and ARMEXEC is like the DOS EXEC function except that it loads and runs a program on the ARM instead of the 8088.

The rest of the functions are all to do with data transfer between the two memory spaces of the PC and Springboard. Data can be transferred either a byte at a time or in blocks, and in both directions. For byte transfers the interface software maintains its own standard input and output buffers, but for block transfers the programmer must supply buffers, passing their length and start address in registers. Both synchronous and asynchronous block transfers are supported.

In a synchronous block transfer both programs, on the PC and the Springboard, must cooperate. The PC issues an ARMBKR (read) or ARMBKW (write) call and the ARM must issue an OSFILE call. If either party is not ready to transfer, the other will be forced to wait until it is; the PC can use ARMPOLL to see if the ARM is ready.

Asynchronous block transfers using ARMBKGET and ARMBKPUT are more passive affairs in which the PC alone transfers data to or from the Springboard. A program running on the Springboard is not expected to do anything at all, and indeed there need not even be a program running on the ARM. This is the fastest way to transfer data, and the calls always succeed.

On the Springboard side of the fence, operating system services are called using OSWORD calls reminiseent of those on the BBC Micro. The ones of partieular interest here are OSWORD 128 which requests a DOS software interrupt, and OSWORD 129 which returns the address of a 64byte parameter block in the PC for those DOS calls, like the filing system ones, that pass parameters in memory. OSFILE saves or loads data to disk (or to the PC memory), and as just mentioned is used to fulfil the ARMs end of a synehronous block transfer.

Performance

I was supplied with Fortran 77 and ARM C compilers and a disk version of Basie V. This latter looks identical to the language used on the Archimedes, and the redundant graphics commands simply do nothing. I ran the PCW Benchmarks, apart from GrafScrn, in Basic V and ARM C with the results shown on page 119. As you will see, the maths functions run even faster than on the Archimedes (TrigLog is twice as fast) while the screen I/O runs at typical IBM PC speeds, exactly as you would expect.

What I didn't expect, however, was the extraordinarily fast time for Store H/D, which must be pushing my old hard disk close to its hardware limits; you can see how slow BasicA (Store H/D=12.7 sees) really is at I/O.

Prices

Springboard 1Mbyte	£1000
Springboard 2Mbyte	£2000
C, Fortran, Pascal, Lisp, Prolog	£250 eaeh
Basic V	£50

Conclusion

At £1-2000 the Springboard is not cheap, and many people will immediately compare its price unfavourably with that of the Acorn Archimedes itself, which starts at £799. When doing this comparison though, don't forget that that Archimedes price buys only 512k of memory, and that the price doesn't include a monitor. A fairer comparison would perhaps be with other fast coprocessor boards for the IBM PC, such as the 80386, the 68020 and the Transputer boards. The prices are broadly similar or more expensive.

Viewed as a highperformance addon for a PC, the Springboard works very well indeed. The interface software is better thought out than most I have seen in this field and makes it quite straightforward to write PC programs that use the ARM as an aceelerator. When the floating point and IEEE podules are released, the Springboard will turn a PC or clone into a very powerful workstation for scientific number-crunching and laboratory control applications.

Acorn Computers is on (0223) 245200.

Dick Pountain is a consultant editor to *PCW,* and has just cowritten a book on occam programming (see the review in this month's Bibliofile). END