PDP-11

DATE: October 22, 1969

SUBJECT: Tentative Design Goals for 11/20 Successors

TO: 11 Steering Committee
John Jones
Alan Kotok
Gordon Bell
Jim O'Loughlin
Nathan Teicholtz
Ad van de Goor

FROM: Bruce Delagi

I am currently working toward the design of the next 11 processor. The goals for this processor are not now well defined and I'd like to avoid building into a corner by at least considering the design goals and implementation for the top of the 11 line - whatever that is. I would appreciate comments any of you might have concerning the goals listed below in relation to:

a. How you would break these goals into those suitable for processors selling (including TTY, 4K core and power supply) for:
$13K - 17K
$17K - 30K
$30K - 50K
$50K - 75K

b. Additional goals you would set for processors in any of the above price classes.

c. Specific suggestions you may have on the realization of these goals.

Tentative design goals for PDP-11 family elements:

1. Reduction of the processor overhead required in specifying a memory access to approximately 400 ns.

2. Overlapping processor operations with memory access.

OCT 24 1969

3. Achieving a speed advantage for subroutines resident in read only memory - as an alternative to microprogramming.

4. Access of processor adders and control logic from the bus in an NPR transfer. This would enable use of the processor logic for boxes that do block arithmetic of the form: $\sum A_i b_{i+j}$ including on-line convolution (digital filtering) convolution (EEG and seismic analysis), and fourier analysis.

5. Overlapping processor usage by two simultaneous processes (while one process is waiting for memory, the other is using the addres and control logic) - Potentially a particularly powerful way to implement block arithmetic processes.

6. Providing bus parity checking - perhaps on both address and data with the option of either bringing the system to a halt or trapping. Parity should be provided on byte operations as well as word. The implication might be that core storage with parity on the 11 is 18 bits wide.

7. Providing for correct bus operation notwithstanding 2 or 3 unpowered control units on the bus. (loading and pass-the-pulse problems)

8. Buffering of the I/O to remote controls (bidirectional buffer).

9. Separation of the memory bus from the I/O bus (to provide a faster, wider path).

10. Multiport memory operation.

11. Overlapping memory operation in banks.

12. Accessing memories in the middle of read/modify/write operations.

13. Using multiple processors on the bus.

14. Sharing devices between busses (e.g. multiported disks).

15. Paging in a TS environment - Two page sizes: 8 bytes for the I/O area (low-overhead user I/O), 512 bytes for other core storage (lease area for each user is 256 bytes for core leases and 256 bytes for I/O leases - a total of 1 page of leases/user).

16. Recovery from "page-out" (page not in memory) faults - i.e. allowing the instruction causing a page out to resume after fetching the page into core.

17. Allowing direct user control of block transfer peripherals - How does the user know how to set the XM bits of the status word? How does the monitor protect other user's core? How does the monitor know when it can roll a user out of core if he has block I/O in progress? Maybe this can not be a direct operation.

18. Attaching and detatching special device handlers to the running monitor as required - billing the user requesting such attachment.

19. Monitor to user communication thru the paging box- paged executes?

20. User to monitor communication - monitor calls with address calculation.

21. Context switching in a guaranteed latency environment - minimum latency on the order of 500 us.

22. Direct user I/O - how direct? Protection from devices requesting at too fast a rate.

23. Priority Sliding - automatic priority reduction with cpu usage time in user interrupt service routines.

24. CPU usage clock, accounting clock.

25. Monitor registers as distinct from user registers. (stack pointer especially)

26. Repeat and Execute instructions (conditional repeat).

27. Limit registers for stack overflow protection.

28. Signed Multiply (16 bit*16 bit to 32 bit) and signed Divide (32 bit/16 bit to 16 bit and 16 bit remainder) watch out for stacks.

29. Multiple word, Multiple place shifts and rotates.

30. Normalize (32 bit) operation.

31. Other 32 bit arithmetic?

32. Floating point operations (normalized, unnormalized; with rounding, without rounding).

33. Floating and double precision register mapping (onto registers and onto core).

34. Minimizing mode switching.

35. Programmed Interrupt Requests (like 9's API).

36. Address match trap console option (causes trap or halt on attempts to reference or write specified (programmable) address).

37. Dynamic examine and modify console operations (examine/modify while processor is running).

38. Execution of instruction specified in console.