**2. What was the elementary base of a computer/processor (relays, vacuum tubes (aka thermoionic valves), transistors, magnetic cores, integrated circuits (ICs) – were they hybrid, low scale integration, Large Scale Integration (LSI), Very Large Scale Integration (VLSI), modern microprocessors)? What where the physical characteristics of the equipment (weight, size, power consumption)?**

**Kokia buvo elementinė kompiuterio/procesoriaus bazė (relės, vakuuminės lempos, tranzistoriai, magnetinės šerdys, integriniai grandynai (IC) – ar jie hibridiniai, mažo integracijos masto, Didelio integracijos masto (LSI), labai didelio integracijos masto (VLSI), ar tai buvo monokristaliniai šiuolaikiniai mikroprocesoriai)? Kokios buvo fizinės įrangos savybės (svoris, dydis, energijos suvartojimas)?**

**3. What type of architecture did both computers have? Were they accumulator based, register based, stack based, memory-to-memory or some other architectures?**

**Kokio tipo architektūrą turėjo abu kompiuteriai? Ar jie buvo akumuliatoriniai, registriniai, stekiniai (dėklo architektūros), atmintis-į-atmintį architektūros arba kokios nors kitos architektūros?**

**4. Were they zero-address, one-address, two-address, three-address or four address machines?**

**Ar tai buvo beadresinės (stekinės), vieno adreso, dviejų adresų,trijų adresų ar keturios adresų mašinos?**

**5. What were the registers in both architectures? Did they have registers at all? Were they general purpose registers or specialized ones? How many registers did each architecture have? What were the widths of these registers?**

**Kokie buvo registrai abiejose architektūrose? Ar šios architektūros iš viso turėjo registrus? Ar tai buvo bendrosios paskirties registrai, ar specializuoti registrai? Kiek registrų turėjo kiekviena architektūra? Kokie buvo šių registrų duomenų pločiai? Kokia buvo specifinė registrų paskirtis?**

**6. Were the flags used in the architectures? What flags were there?**

**Ar požymių bitai buvo naudojami šiose architektūrose? Kokie požymiai buvo naudojami?**

Naudojami. The Program Status includes six processor flags as follows: Carry (C), Zero (Z), Sign (S), Parity/Overflow (P/V), Decimal Adjust (D), and Half Carry (H).

**7. What was the data width (machine word) of each architecture?**

**Koks buvo kiekvienos architektūros duomenų plotis (mašininis žodis)?**

**8. What was the memory layout of each system? Was address space continuous, or was it segmented, paged, subdivided into banks? What was the (effective) width of the address? What was the maximum possible amount of memory in each system? What was the typical amount of memory with which the system was used?**

**Koks buvo kiekvienos sistemos atminties išdėstymas? Ar adresų buvo erdvė ištisinė, ar ji buvo suskirstytas į segmentus, puslapius, atminties į bankus? Koks buvo (efektyvus) adreso plotis? Koks buvo maksimalus įmanomas atminties kiekis kiekvienoje sistemoje? Koks buvo tipiškas atminties kiekis, su kuria sistema buvo naudojama?**

**9. Was virtual memory supported, and how? Was it paged or segmented?**

**Ar buvo palaikoma virtualioji atmintis ir kaip? Ar virtuali atmintis buvo realizuoti, naudojant puslapiavimą, segmentavimą, abu šiuos mechanizmus?**

**10. What was the ISA of each architecture? How many instructions did each architecture have? What were the classes of the instructions? What instructions formats did each architecture support? Provide examples of some 8-16 instructions. What instructions were similar in both architectures? Which instructions were different?**

**Kokia buvo kiekvienos architektūros komandų sistema (ISA)?** Kiek mašinos komandų turėjo kiekviena architektūra?Kokios buvo instrukcijų (komandų) klasės? **Kokius instrukcijų formatus palaikė kiekviena architektūra?** Pateikite 8–16 instrukcijų pavyzdžius**. Kokios komandos buvo panašios abi architektūros? Kurios komandos skyrėsi?**

(apie 50-60 psl. dokumentacijoj)

Turėjo apie 800 skirtingų komandos kodų.

Komandų klasės:

• Load and Exchange (pvz.: EX, EXB, EXL dst,src – Exchange) (čia įeina ir load, pop, push ir pan.)

• Arithmetic (INDEX, INDEXL dst,sub,src Index), aišku čia ir visos + - \* / compare

• Logical ( XOR XORB XOR dst,src Exclusive Or) (and, or, test)

• Program Control (Trap cc, src Contditional trap) call, jp, jr, enter, exit

• Bit Manipulation (RES RESB RES dst,src reset bit), set Bit, bit test

• Bit Field (INSRT dst, src, pos, siz Insert field)

• Rotate and Shift (RlC RlCB RlCl dst,src Rotate left through Carry)

• Block Transfer and String Manipulation (CPD CPDB CPDL dst,src,r,cc Compare and Decrement)

• Input/Output (INIR INIRB INIRL dst,src,r Input, Increment and Repeat)

• CPU Control (SETFLG flag Set Flag)

• Extended Instructions (The CPU supports four types of extended instruc tions: EPU internal operations that do not require any data transfer; transfer of one to six teen words of data between the EPU and consecutive word or longword general-purpose registers; transfer of one byte of data between the EPU and the flag byte of the FCW; and the transfer of one to sixteen bytes or words of data between the EPU and memory. The flags are affected only when the flag byte is loaded.)

**11. What were the addressing modes supported by each architecture? Which modes were similar, and which were different?**

**Kokius adresavimo būdus palaikė kiekviena architektūra? Kurie režimai buvo panašūs, o kurie skyrėsi?**

**12. What were the I/O capabilities of each architecture? Kokios buvo kiekvienos architektūros I/O galimybės?**

Komandos: Input; Input and Decrement; Input, Decrement and Repeat; Input and Increment; Input, Increment and Repeat; Output, Decrement and Repeat; Output, Increment and Repeat; Output; Output and Decrement; Output and Increment

**13. Were interrupts supported for each architecture? How was the interrupt support similar, and how was it different in both architectures? Ar buvo palaikomi pertraukimai? Kuo pertraukimų mechanizmai buvo panašūs, kuo jie skyrėsi abiejose architektūrose?**

267 ir 293 psl dokumentacijoj

Ths CPU recognizes three kinds of interrupt sig nslled on separsts pinal non-masksble, vectored, snd non-vectored. Non-mssksble interrupts sre slwsys enabled. Vsctored and non-vectored inter rupta can be selectively enabled by bita VIE snd NVIE in the FCW.

An interrupt occurs when an enabled interrupt request is signalled on a CPU pin. The CPU gener ates an interrupt acknowledge trsnsaction on tha external intarface to fetch the idantifier word, which is then savad on the system stack. For vec tored interrupts, the low-order byte of the iden tifier word is used to select a pointer to a par ticular interrupt handler routine. Refer to Sec tion 8.7.5 for more details about interrupt request and acknowledge.,

**14. What data types did each architecture support on instruction level? Was fixed point or floating point supported by hardware and how? Where integers sign-magnitude, one's complement, two's complement? What other "exotic" data types did architecture support (decimal numbers, complex numbers, etc.)?**

**Kokius duomenų tipus palaikė kiekviena architektūra aparatūros lygyje? Ar buvo palaikoma fiksuoto kablelio, slankiojo kablelio aritmetika? Ar sveikieji skaičiai buvo koduojami kaip ženklas-dydis, kaip vieneto papildinys (atvirkštinis kodas), dvejeto papildyti (papildomas kodas)? Kokius kitus „egzotiškus“ duomenų tipus palaikė architektūra (pvz. dešimtainius skaičius, kompleksinius skaičius ir kt.)?**

Data formats: The CPU manipulates bits, bytes (a bits), words (16 bits), longwords (32 bits), and quadwords (64 bits) of data. Within a byte, word, longword, or quadword, the bits are numbered from right to left, from least to most significant (Figure 2-1). This is consistent with the convention that bit n corresponds to position 2n in the represen tation of binary numbers.

Iš komandų dalies: The arithmetic group consists of instructions for performing integer arithmetic. The basic instruc tions operate on unsigned binary integers or signed twos complement binary integers. Support is provided for Binary Coded Decimal (BCD) arith metic and multiple precision arithmetic. Add and Subtract operate on bytes, words, or longwords. The Multiply instructions operate on words or long words and compute a double-precision product. The Divide instructions operate on words or longwords, using a double-precision dividend.

**15. What was the speed of each system? How did clock frequencies, clock cycles per instruction, instruction rates compare? Which system was faster?**

**Kokia buvo kiekvienos sistemos greitaveika? Kokie buvo taktinių generatorių dažniai, vidutinis/mažiausias/didžiausias ciklų skaičius, reikalingas kiekvienai komandai įvykdyti, vidutinė sistemos greitaveika? Kuri sistema buvo našesnė? Koks buvo kainos ir našumo santykis?**

**16. Did the architectures use cache memory? If yes, how much? Ar architektūros naudojo spartinančią atmintį? Jei taip, kokio dydžio?**

276psl.

The CPU implements a cache mechanism that keeps a copy of recently used memory locations on-chip. These locations can contain both instructions and data. On memory fetches, the CPU examines the cache to determine if the addressed information is stored there. If the information is in the cache (a hit), then the CPU fetches the copy from the cache, and no transaction is necessary on the external interface. If the information is not in the cache (a miss), then the CPU performs a memory read transaction to fetch the missing information and stores a copy of the information into the cache, replacing the least recently used data in the cache. Thus, the cache serves to reduce the number of memory read transactions, providing a substantial boost to performance. Software can control the cache mechanism in several ways. The System Configuration Control Longword register contains separate control bits (CI and CD) that enable the cache for instruction and data references and another bit (CR) that enables the cache replacement algorithm. In page table entries, the NC bit can be set to disable the use of the cache for selected pages. The Purge Cache instruction can be executed to invali date the contents of the cache when a memory loca tion that may have been copied into the cache has been modified by another processor. For example, if a slave processor reads from a peripheral port to a memory location that may be copied in the cache, the cache must be purged. Similarly, if two or more tightly-coupled CPUs can alternately execute one process, the cache must be purged when the operating system changes from executing one user-process to another. Appendix C describes the cache mechanism in more detail, control and interface.

**17. What were the typical application areas of each architecture? How were they used? Describe briefly (on paragraph) one particular installation of each architecture.**

**Kokios buvo tipinės kiekvienos architektūros taikymo sritys? Kaip šios architektūros buvo naudojamos? Trumpai apibūdinkite (vienoje pastraipoje) vieną konkretų kiekvienos architektūros panaudojimo pavyzdį.**

**18. How much software was written for each discussed architecture, is it (still) available, what application domains did it target? What compilers and programming tools (debuggers, profilers, assemblers) did architectures have? What software libraries were available?**

**Kiek programinės įrangos buvo parašyta kiekvienai aptariamai architektūrai, ar ji (vis dar) prieinama, kur ji buvo naudojama? Kokie buvo prieinami kompiliatoriai ir programavimo įrankiai (derintojai, profiliuotojai, surinkėjai)? Kokios programinės įrangos bibliotekos buvo prieinamos?**

**19. (optional) are there emulators available for both architectures? If you find some, provide their URLs and/or publication metadata.**

**(neprivaloma) ar yra emuliatorių abiem architektūroms? Jei tokių rasite, pateikite jų URL ir (arba) leidinio metaduomenis.**