**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?**

32 bendros paskirties registrai

Thirty-two 32-bit general registers provide the central resource for all computation (Figure 2-6). They are numbered GR 0 through GR 31, and are available to all programs at all privilege levels. GR 0, GR 1, and GR 31 have special functions. GR 0, when referenced as a source operand, delivers zeros. When GR 0 is used as a destination, the result is discarded. GR 1 is the implied target of the ADD IMMEDIATE LEFT instruction. GR 31 is the instruction address offset link register for the base-relative interspace procedure call instruction (BRANCH AND LINK EXTERNAL). GR 1 and GR 31 can also be used as general registers; however, software conventions may at times restrict their use. Implementations must provide seven registers called shadow registers, numbered SHR 0 through SHR 6, into which the contents of GRs 1, 8, 9, 16, 17, 24, and 25 are copied upon interruptions. The contents of these general registers are restored from their shadow registers when a RETURN FROM INTERRUPTION AND RESTORE instruction is executed. In systems which support virtual memory, eight space registers, numbered SR 0 through SR 7, contain space identifiers for virtual addressing. Instructions specify space registers either directly in the instruction or indirectly through general register contents.

There are twenty-five defined control registers, numbered CR 0, and CR 8 through CR 31, which contain system state information. CR 11, the Shift Amount Register, is readable and writable by code executing at any privilege level. CR 16, the Interval Timer, is readable and writable only by privileged software, unless the PSW S-bit is 0, in which case it is readable by code executing at any privilege level. CR 26 and CR 27, two of the temporary registers, are readable by code executing at any privilege level and writable only by code executing at the most privileged level. All other defined control registers are accessible only by code executing at the most privileged level. The control registers are shown in Figure 2-11 and described in the following sections. Moving from control registers into general registers copies the register right aligned into the general register. Moving to control registers from general registers copies the entire general register into the control register. Control registers 1 through 7 are reserved registers, and the unused bit positions of the PIDs and the Coprocessor Configuration Register are reserved bits. The unused bits of the Shift Amount Register are nonexistent bits. In Level 0 systems, CRs 8, 9, 12, 13, 17, and 20 are nonexistent registers.

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

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

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

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

Instruction size is fixed length and fixed format — one word (32-bit), which facilitates pipelining

**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?**

*Translation Lookaside Buffer* is a hardware structure doing virtual-to-physical memory address translations which takes virtual page numbers and returns the corresponding physical page number. PA-7000 is the last PA-RISC processor with seperate instruction and data TLBs, all later PA 1.1 and 2.0 CPUs use combined TLBs while older PA-RISC 1.0 processors use huge TLBs (even for today’s standards):

* *PA 1.1*: If a virtual address has to be translated to a physical address, the corresponding TLB is searched for an entry matching the Virtual Page number. If an entry is found, the 20-bit Physical Page number, delivered by the TLB, is concatenated with the original 12-bit page offset to the build up the 32-bit absolute physical address.
* *Hardware*: If the CPU implementation provides a hardware TLB miss handler, it attempts to find the virtual-to-physical translation in the *Page Table*. If successful, the translation and protection fields are inserted in the TLB. If not successful, an interruption occurs so the software miss handler can complete the translation.
* *Software*: If software TLB miss handling is implemented, a TLB miss fault interruption routine performs the translation. It inserts the translation and protection fields in the TLB and afterward restarts the interrupted routine, in which the TLB miss occurred.

Similar to the TLB, the BTLB provides virtual-to-physical address translations. The BTLB however maps large address ranges rather that single pages as the TLB. These large address ranges are block translations and therefore stored in the Block Translation Lookaside Buffer. These block translations are useful for virtual address ranges that do not get paged in or out.

BTLBs were only implemented on 32-bit PA-RISC processors (PA-7x00), 64-bit PA-RISC instead implemented variable page sizes, thus any entry can be of >4k mapping.

**10. What was the ISA of each architecture? What instructions formats did each architecture supportz?**

**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**.**

For bundling purposes instructions are divided into classes:

|  |  |
| --- | --- |
| ***PA-RISC superscalar instruction classes*** | |
| **Class** | **Description** |
| **FLOP** | **Floating point operation** |
| **LDST** | **Loads and stores** |
| **ALU** | **Integer ALU** |
| **MM** | **Shifts, extracts, deposits** |
| **NUL** | **Might nullify successor** |
| **BV** | **Branch Vectored (BV) local, Branch (BE) external** |
| **BR** | **Other branches** |
| **FSYS** | **FTEST and FP status/exception** |
| **SYS** | **System control instructions** |

**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?**

Only three addressing modes: long/short displacement and indexed

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

**Memory and I/O Controller (MIOC)**

The *Memory and I/O Controller* (MIOC) in the [PA-7100LC](https://www.openpa.net/pa-risc_processor_pa-7100lc.html) and [PA-7300LC processor](https://www.openpa.net/pa-risc_processor_pa-7300lc.html) integrates DRAM, cache and I/O controllers onto the processor die. MIOC is similar on both CPUs, with the PA-7300LC MIOC having wider data paths to L2 cache and RAM and supporting the advanced GSC+ bus over the older GSC on PA-7100LC.

MIOC’ integrated memory controller requires only buffers and DRAM modules to build up complete memory subsystem. The [PA-7300LC](https://www.openpa.net/pa-risc_processor_pa-7300lc.html) MIOC memory controller includes a Second Level Cache Controller SLC, which provides an optional L2 cache, ranging from 32 KB to 8 MB. It shares the data bus with the DRAM subsystem, so it has the same width and same optional SEDC error control.

* Execution units and internal caches attach on-chip to the MIOC
* External cache, L1 on PA-7100LC, L2 on PA-7300LC, attach to MIOC via 64-bit or 128-bit
* Memory attaches to MIOC via 64-bit on PA-7100LC or 128-bit on PA-7300LC
* GSC, the system main bus, attaches to MIOC
* Support for 4, 16, 64 and 256 Mbit modules, FPM and EDO DRAM at 3.3 or 5.0 V
* Up to 16 physical memory slots
* Support for a wide range of core frequencies

**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?**

**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.)?**

Floating Point Unit (FPU)

The *Floating Point Unit* is an assist processor logically added to a system to improve the performance on floating-point operations. The processor can be on a seperate chip (*e.g.,* [PA-7000](https://www.openpa.net/pa-risc_processor_pa-7000.html)) or integrated onto the central CPU die (all PA-RISC CPUs upwards). The FPU executes special floating point instruction to perform arithmetic on its own set of independent registers (*register file*) and to move data between its own registers and the system’s lower memory hierarchy. The FPU execution stage is pipelined. All PA-RISC FPUs contain thirty-two 64-bit registers, which can also be used as sixty-four 32-bit registers and sixteen 128-bit registers.

Duomenų tipai:

Bitai, baitai,

A screenshot of a computer

Description automatically generated

Floating-Point Numbers The binary floating-point number representation conforms to the ANSI/IEEE 754- 1985 standards. Single-word (32-bit), double-word (64-bit), and quadruple-word (128-bit) binary formats are supported. Single-precision floating-point numbers must be aligned on word boundaries. Double-precision and quad-precision numbers must be aligned on doubleword boundaries. See Chapter 6, “Floating-point Coprocessor”, for detailed information on the floating-point formats. Packed Decimal Numbers Packed decimal data is always aligned on a word boundary. It consists of 7, 15, 23, or 31 BCD digits, each four bits wide and having a value in the range of 0x0 to 0x9, followed by a 4-bit sign as shown in the following figure

**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?**

**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.**

Reduced instruction set (simple, efficient instructions)

Instruction set is implemented in hardware (hardwired) and not microcoded.

HP designed a few beyond RISC features into PA-RISC: very large address space (48-bit direct addressing for 64-bit virtual), split instruction and data caches, high-bandwidth internal buses, memory mapped I/O for high-speed devices of the future, precision interrupts for real-time response and optimized I/O bus interfaces to enable easy bus converters.