1 This question concerns parallelism in the Intel Itanium 2 processor, as described in the paper “Itanium 2 Processor Microarchitecture” (McNairy and Soltis, IEEE Micro March-April 2003), which you should have available to you in the examination. **See, in particular, pages 46, 50 and 54.** Where the paper is incomplete, you are invited to speculate using your understanding of the underlying architectural principles.

a In a proposed dual-core Itanium 2 design, two CPUs are integrated on the same die. Should they share the L2 cache? Identify at least four issues.

b In a proposed simultaneous multi-threading (SMT, also known as hyperthreaded) Itanium 2 design, a single CPU core is extended with two program counters, and two register sets, so that it can execute two different threads at the same time. Referring to Figure 1 (page 46), identify which parts of the design would have to be changed, and explain briefly what would have to be done.

c Under what circumstances might an SMT Itanium 2 design achieve higher performance than the actual Itanium 2?

d Under what circumstances might an SMT Itanium 2 design achieve lower performance than the actual Itanium 2?

The four parts carry, respectively, 40%, 30%, 20%, and 10% of the marks.

2 This question concerns memory hierarchy in the Intel Itanium 2 processor, as described in the paper “Itanium 2 Processor Microarchitecture” (McNairy and Soltis, IEEE Micro March-April 2003), which you should have available to you in the examination. **See, in particular, pages 45 and 52–54.** Where the paper is incomplete, you are invited to speculate using your understanding of the underlying architectural principles.

a The Itanium 2’s L1D TLB has 32 entries. The 16KB L1D cache is four-way set-associative with 64-byte lines and prevalidated tags. How many bits are occupied by tags?

b A conventional alternative to this L1D cache design would be a virtually-indexed, physically-tagged cache (physical addresses are 50 bits), accessed concurrently with a TLB. How many bits need to be compared to identify a cache hit? How many bits have to be compared in the Itanium 2’s prevalidated tags design to identify a cache hit?

c Why did the Itanium 2 designers build separate level 1 and level 2 caches, instead of just building one big level 1 cache? Give at least two reasons.

d Why did the Itanium 2 designers build separate level 2 and level 3 caches, instead of just building one big level 2 cache? Give at least two reasons.

e Why is the L3 cache designed to support a higher rate of access to tags than to data?

f Why is the level 1 instruction cache designed to support higher rate of access to tags than to data (access to tags is dual-ported)?

The six parts carry, respectively, 15%, 20%, 20%, 20%, 10%, and 15% of the marks.

3 This question concerns speculation in the Intel Itanium 2 processor, as described in the paper "Itanium 2 Processor Microarchitecture" (McNairy and Soltis, IEEE Micro March-April 2003), which you should have available to you in the examination. **See, in particular, page 51.** Where the paper is incomplete, you are invited to speculate using your understanding of the underlying architectural principles.

a Although not discussed in the IEEE Micro paper, the Itanium 2 has a speculative load instruction, "`ld.s rD=[rS]`" (where register `rS` holds the load address, and `rD` is the destination). It is designed to be used when a load's result may not actually be needed. The "`chk.s rD,recovery`" instruction checks that the value in `rD` is actually valid, and branches to the label `recovery` if it is not.

i) Use a *small* example to illustrate why a speculative load instruction might be useful.

ii) Each of the Itanium 2's general purpose and floating-point registers has an additional "Not a Thing" (NaT), used to mark a value that depends on an invalid speculative instruction. Under what circumstances should a floating-point divide instruction produce a NaT result?

b The Itanium 2 also has an "advanced-load" instruction `ld.a rD=[rS]`" (where register `rS` holds the load address, and `rD` is the destination). This variant uses the Advanced Load Address Table (page 51 of the IEEE Micro article).

i) Use a *small* example to illustrate why an advanced-load instruction might be useful.

ii) Explain what happens if the advanced-load is invalidated.

iii) What happens if more than 32 different advanced load instructions are issued?

c Why is `ld.a` (advanced load) different from `ld.s` (speculative load)? Explain how they differ, and why.

d Would advanced load and speculative load instructions be useful in a dynamically-scheduled processor with a re-order-buffer (ROB) to support speculative execution? Explain your answer carefully.

The four parts carry, respectively, 25%, 40%, 15%, and 20% of the marks.

4 a You are designing a disk array which will handle 1200 lookup requests per minute on average. The mean response-time requirement for the array is that all requests should be answered within 0.01 seconds. What is the mean number of requests that should be in the array if the response-time requirement is to hold?

b You are running a web server, from which, empirically, you have found that the mean number of web page requests in the system is:

$$\frac{2+\rho^2}{2(1-\rho)} - \rho$$

where ρ is the server utilisation. You would like to fix this quantity to be exactly L requests, so that you can make better use of memory resources. What limitations are there on the value of L with this buffer model? What does this suggest about the usage of the web server?

c Given a web-graph, G, representing the link structure of the internet and a matrix P defined by $P_{ij} = 1/\deg(u_i)$ if a link exists from page u_i to u_j in G and 0 otherwise. P' is created by adding P to a second matrix D. D is defined to be the product of $\vec{d}$ and the personalisation vector $\vec{p}$, where $d_i = 1$ if $\deg(i) = 1$ and 0 otherwise. The rows of a third matrix E consist entirely of the personalisation vector $\vec{p}$. The modified transition matrix, A, is given by $A = cP' + (1-c)E$.

i) Explain the roles of the matrices D and E in the above description.

ii) Given a set of m bookmarks $B \subset G$ and a homepage $h \in G$, $h \notin B$. Define the personalisation vector $\vec{p}$ which represents a uniform probability for visiting any of the bookmarks or the homepage.

iii) The definition of a single PageRank iteration is $\vec{x}_{(k+1)} = \vec{x}_{(k)}A$, whereas this iteration is actually implemented as:

$$\vec{x}_{(x+1)} = c\vec{x}_{(k)}P + (1 - c||\vec{x}_{(k)}P||_1)\vec{p}$$

Why is the iteration not implemented as initially defined, using the A matrix? By considering the number of additions and multiplications used to calculate the second equation, ascertain the complexity of the implementation.

The three parts carry, respectively, 10%, 25%, and 65% of the marks.