

Norwegian University of Science and Technology (NTNU)
DEPT. OF COMPUTER AND INFORMATION SCIENCE (IDI)


Course responsible: Professor Lasse Natvig
Quality assurance of the exam: Associate Professor Magnus Jahre
Contact person during exam: Lasse Natvig, Phone: 906 44 580

Deadline for examination results: 27th of June 2011.

EXAM IN COURSE TDT4260/DT8803 COMPUTER ARCHITECTURE

Monday 6th of June 2011

Time: 1500 – 1900


<p>This course is part of EMECS: Erasmus Mundus Master Program in Embedded Computing Systems. The students are free to answer the exercises in English or Norwegian (Bokmål or Nynorsk).</p>

Supporting materials: No written and handwritten examination support materials are permitted. A specified, simple calculator is permitted.

By answering in short sentences it is easier to cover all exercises within the duration of the exam. The numbers in parenthesis indicate the maximum score for each exercise. We recommend that you start by reading through all the subquestions before answering each exercise.

The exam counts for 80% of the total evaluation in the course. Maximum score is therefore 80 points.

Exercise 1) Measuring computer performance (Max 14 points)

a) (Max 6 points) The increase in processor performance on integer operations was quite stable in the period 1985 to 2002, but has changed after 2002. Explain briefly how the increase has been since 2002, and explain some of the reasons that have given the change.

Solution sketch:

(a) The increase became lower from around year 2002 and has been approximately 20% pr. year (It was over 50% per year before).

(b) * i) Heat problems; it is limited how much power (heat) a normal air-cooled chip can dissipate. * ii) Limited ILP (Instruction Level Parallelism), i.e. it became difficult/impossible to exploit ILP to a larger extent than what was currently done in single core processors. * iii) Relatively very slow improvement in memory-latency.

b) (Max 4 points) Give a general (informal if you like) definition of each of the two concepts bandwidth (throughput) and latency (response time).

Solution sketch: **Bandwidth** is the total amount of “work” processed within a given time period, while **latency** (response time) is the time from the start (initiation) of an operation to is has been completed. (From lecture 1 slide 35: • Bandwidth: number of events per unit time (e.g., M bits/second over network, M bytes / second from disk. • Latency: elapsed time for a single event (e.g., one-way network delay in microseconds, average disk access time in milliseconds))

c) (Max 4 points) Explain how bandwidth has changed roughly during the last 20 years compared to latency for primary memory (main memory) and for hard disk technology.

Solution sketch: (a) Primary memory: Bandwidth (Mbit/s) has increased much more (approx. 100x) compared to latency (approx. 5x faster) (b) Hard disk: the same trend: Bandwidth is measured in MB/s for disk transfers, approx. 100x faster, latency approx. 10x better.

Exercise 2) Loop unrolling and VLIW (Max 14 points)

a) (Max 4 points) Describe how you would do loop unrolling in the case where you do *not* know the number of iterations of the loop, called n .

Solution sketch: Choose a value for k , and make k copies of the loop body in an unrolled loop (Called A). Iterate loop A (n/k) times (round down to integer value). Then finish by executing the “reminder”, by the loop B which is $(n \bmod k)$ iterations of the original loop. (You may do B before A, as described in lecture 2, last slide. For large values of n , most of the execution time will be spent in the unrolled loop.)

b) (Max 6 points) Describe briefly the advantages and disadvantages of VLIW. Explain also briefly why it has been a popular architecture in embedded systems.

Solution sketch:

(a) **advantages:** * “Simpler” hardware because the HW does not have to identify independent instructions.

(b) **disadvantages:** * Increase in code size, since parts of the VLIW can be empty, and if loop unrolling is used. * Reduced binary code compatibility: strict VLIW => different numbers of functional units (and unit latencies) require different versions of the code. * Relies on smart compiler * Code incompatibility between generations

(c) **embedded systems:** where hardware simplicity is important, applications exhibit plenty of ILP, and binary compatibility is a non-issue.

c) (Max 4 points) 1st generation VLIW systems operated in lock-step without hazard detection.

Describe the main problem/disadvantage of this choice, and why caches make this problem larger.

Solution sketch: A stall in any functional unit pipeline causes entire processor to stall, since all functional units must be kept synchronized. Compiler might predict function units, but caches hard to predict. Modern VLIWs can to some extent avoid this by identifying dependences between bundles and stall.

Exercise 3) Caches and prefetching (Max 20 points)

a) (Max 4 points) Explain briefly why most computers today have a so-called memory hierarchy.

Solution sketch: The memory hierarchy is a result of the increasing gap between speed of processors compared to the response time (latency) of memory. We want a large memory, but a large memory is for technical reasons slower than smaller memories (caches). Therefore, we store the data and instructions that are most frequently used close to the processor in small and fast memories. This can be organised in different levels; Register File ==> Cache level 1 (L1) ==> Cache level 2 (L2) etc.

b) (Max 6 points) When measuring cache miss the misses can be categorized as “four C’s”. Two of these are Compulsory and Coherence. Explain briefly what kind of misses each of them represent. Give also the term for the two other C’s, and explain what kind of misses they represent. **Solution sketch:** **Compulsory:** When you start a computer/processor its cache is empty. Every time data or instructions are accessed we get a miss, they have to be fetched from memory. These compulsory misses are also called cold-start misses, and they will happen also for an infinitely large cache. **Coherence** denotes those misses we get in the cache since the cache-coherence protocol sometimes have to invalidate cache copies in a multiprocessor when another processor gets write access to the data. In addition, **capacity** misses denotes those we get since cache blocks (lines) are replaced when a new block is fetched to the same position in the cache. These could have been omitted if the cache had more room. **Conflict** misses occur in a cache that is not fully associative. Here different addresses can be mapped to the same cache-block or set, and we get a collision, and a block is replaced.

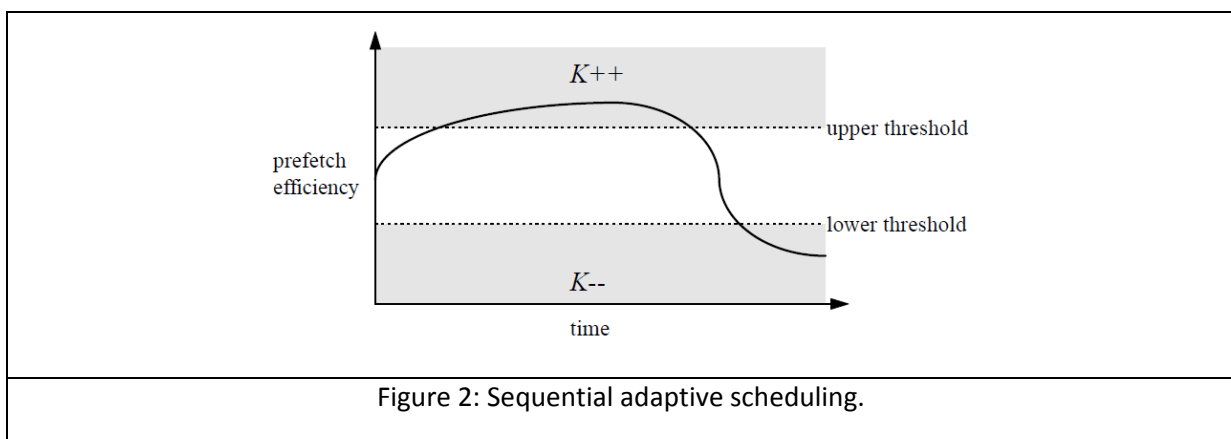
c) (Max 6 points) Prefetching schemes can be described by answering three basic questions starting with “When”, “Where” and “What”. Discuss these three questions/dimensions and give an example showing that these three questions are not independent of each other.

When to prefetch? Can be initiated explicitly by a fetch operation in a program or by a HW-prefetching mechanism, or a combination. Must be done at the right time. If too early, it can displace other data that has not been used yet, or it can have been replaced by other data before it was used by the processor. If the prefetch is issued too late, it may not arrive before the actual memory.

Where to put the data? Must be moved into a higher level of the memory hierarchy to provide a performance benefit. In some type of cache memory, L1, L2? Some schemes place prefetched data in dedicated buffers to protect the data from premature cache evictions and prevent cache pollution.

What to prefetch? Data can be prefetched in units of single words, cache blocks, contiguous blocks of memory, or program data objects. The amount of data fetched is also determined by the organization of the underlying cache and memory system. Cache blocks may be the most appropriate size for uniprocessors and SMPs, while larger memory blocks may be used to amortize the cost of initiating a data transfer across an interconnection network of a large, distributed memory multiprocessor.

d) (Max 4 points) Dahlgren et.al. proposed a prefetching policy called *sequential adaptive prefetching*. Explain the policy based on Figure 2, found below.



(Background) K is the prefetching degree. If $K > 1$, and if the block b is a prefetched, we check if the block $b+1, \dots, b+K$ are present in the cache, and if not, they are fetched from memory. Such prefetching of more blocks can reduce the miss rate when we have degree of spatial locality, but also give additional traffic and can pollute the cache with block we will not access.

Dahlgren et al. proposed an adaptive sequential prefetching policy that allows the value of K to vary during program execution in such a way that K tries to follow the degree of spatial locality exhibited by the program, dynamically. The prefetch efficiency metric is periodically calculated by the cache as an indication of the current spatial locality characteristics of the program. It is defined to be the ratio of useful prefetches (results in hit) to total prefetches. K is initialized to one, incremented whenever the prefetch efficiency exceeds a predetermined upper threshold, and decremented whenever the efficiency drops below a lower threshold, as shown in Figure 7. Note that if K is reduced to zero, prefetching is effectively disabled. At this point, the prefetch hardware begins to monitor how often a cache miss to block b occurs while block $b - 1$ is cached, and restarts prefetching if the respective ratio of these two numbers exceeds the lower threshold of the prefetch efficiency.

Exercise 4) Multi-cores architectures (Max 18 points)

a) (Max 4 points) The UltraSparc T1 (Niagara) has a pipeline where some of the subunits are replicated (copied) in 4 copies to make rapid context switches between 4 threads possible. Describe which units were replicated in this way.

Solution sketch: It has 4 instruction buffers (IB), 4 copies of the PC (program counter logic, used to select the right instr. from IB), 4 register files, and 4 store buffers.

b) (Max 4 points) Based on Figure 1, explain briefly the role of the unit called Home Engine and Remote Engine, as well as the Interconnect Links to the left in the figure.

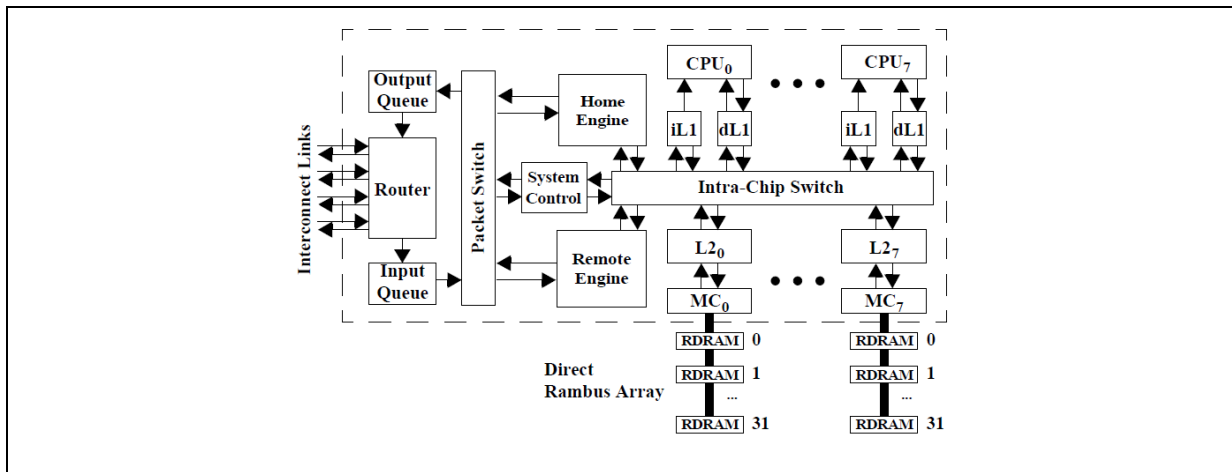


Figure 1: Piranha processing node

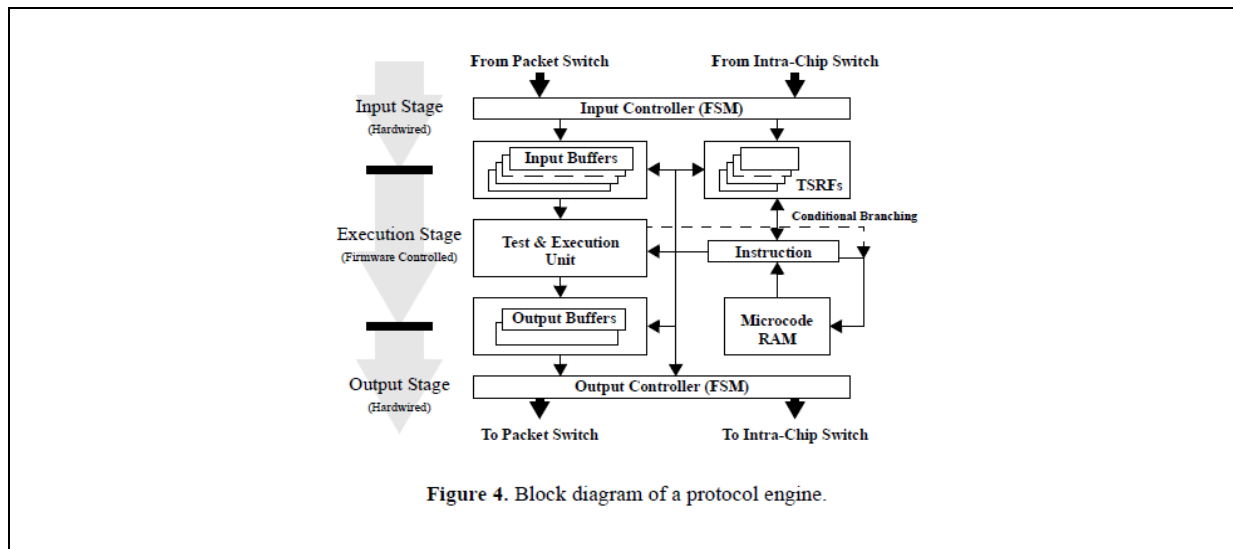
Solution sketch: Connected to the ICS (Intra-Chip Switch) are two protocol engines, the Home Engine (HE) and the Remote Engine (RE), which support shared memory across multiple Piranha chips. The interconnect links are used for connecting such chips into a multi-chip multiprocessor system.

c) (Max 4 points) Describe briefly the role of the Piranha I/O node and how it is different from the processing node.

Solution sketch: (Background: While the Piranha processing chip is a complete multiprocessor system on a chip, it does not have any I/O capability.) The actual I/O is performed by the Piranha I/O chip, which is relatively small in area compared to the processing chip. Each I/O chip is a stripped-down version of the Piranha processing chip with only one CPU and one L2/MC module. The router on the I/O chip is also simplified to support only two instead of four links. (From the programmer's point of view, the CPU on the I/O chip is indistinguishable from one on the processing chip. Similarly, the memory on the I/O chip fully participates in the global cache coherence scheme)

d) (Max 6 points) Describe briefly the protocol engines in Piranha and how they are implemented. Sketch a rough block diagram if you like. (Hint: They have three main stages).

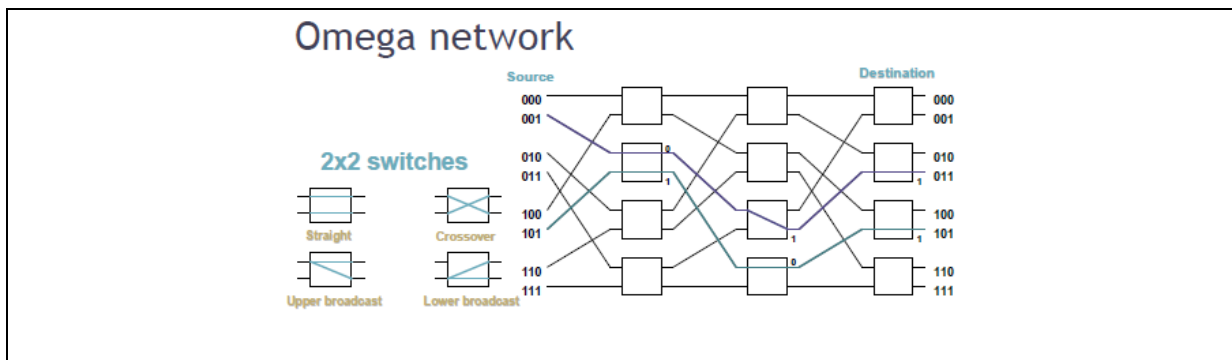
Solution sketch: Implemented as microprogrammable controllers, with the home and remote engines being virtually identical except for the microcode that they execute. Consisting of three independent (and decoupled) stages: the input controller, the microcode controlled execution unit, and the output controller. The input controller receives messages from either the local node or the external interconnect, while the output controller sends messages to internal or external destinations. The microcode memory supports 1024 21-bitwide instructions (the current protocol uses about 500 microcode instructions per engine). The actual protocol code is specified at a slightly higher level with symbolic arguments, and C-style code blocks, and a sophisticated microcode assembler is used to do the appropriate translation and mapping to the microcode memory. (Not all details required for full score)



Exercise 5) Interconnection networks, green computing and HPC (Max 14) points)

a) (Max 6 points) The Omega network is an example of a multistage interconnection network. Explain how such a network is a compromise between a bus and a crossbar, and sketch one such network (e.g. Omega) providing communication from any of 8 nodes to any of the same 8 nodes. (The exact details of the Omega “interconnection-pattern” is not needed)

Solution sketch: The cost of a bus is $O(n)$, but only one communication transfer can be in flight at a given time. The cost of a full cross-bar is $O(n^2)$, and n communication transfers can be processed simultaneously. A multistage network has a cost of $O(n \log n)$, can have n ongoing transfers, but each transfer goes thru $\log n$ stage. See example for Omega-net.



b) (Max 5 points) Your task is to analyze the performance and power efficiency of a parallel program on two different machines. In Machine A, the chip area is used to provide 4 high-performance processing cores where each core can complete 2 units of work each second. In Machine B, the area is used to provide 16 cores that can complete 1 work unit each second. Both machines use the same amount of power. The program is mapped to one thread per core and consists of 32 units of work. Furthermore, it scales ideally and all communication overheads are negligible. What is the runtime of the program on Machine A and B? Which machine is most power efficient for this program?

Solution sketch:

	<i>Machine A</i>	<i>Machine B</i>
<i>Total amount of work</i>	<i>32</i>	<i>32</i>
<i>Work units per core</i>	<i>32 / 4 = 8</i>	<i>32 / 16 = 2</i>
<i>Total run time</i>	<i>8 / 2 = 4 seconds</i>	<i>2 / 1 = 2 seconds</i>

Both machines use the same amount of power. Since Machine B achieves higher performance than Machine A, it is more power efficient.

c) (Max 3 points) GPFS is a filesystem used in supercomputers (HPC systems), also at NTNU. Describe GPFS briefly.

Solution sketch: GPFS parallel file system, ((33 TiB fiber disks 62 TiB SATA disks)). GPFS is geared at high BW I/O, used extensively in HPC and in the database industry. (Disk access is approx. 1000 times slower than memory access, hence key factor for performance are) - spreading (striping) files across many disk units, using memory to cache files, hiding latencies in software. High transfer rates is achieved by distributing files in blocks round robin across a large number of disk units. In addition to multiple disks servicing file I/O, multiple threads might read, write or update (R+W) a file simultaneously. GPFS use multiple I/O servers (4 dedicated nodes on njord), working in parallel for performance, maintaining file and file metadata consistency. (Not all details required)

...---oooOOOooo---...