Universidad Zaragoza

Academic Year/course: 2021/22

30237 - Multiprocessors

Syllabus Information

Academic Year: 2021/22
Subject: 30237 - Multiprocesadores
Faculty / School: 110 - Escuela de Ingeniería y Arquitectura
Degree: 439 - Bachelor's Degree in Informatics Engineering
ECTS: 6.0
Year: 3
Semester: Second semester
Subject Type:
Module:

1. General information

1.1. Aims of the course

The course and its expected learning results respond to the following approaches and objectives:

- To present simple code analysis techniques to determine if data parallelism exists.
- To present a vector processor design (organization) that exploits this data parallelism.
- Flynn's taxonomy. Classification of organizations according to their ability to exploit parallelism.
- Analyze and understand the basic building blocks underlying the design of modern shared-memory multiprocessors.
- Achieve the learning outcomes.

IMPORTANT: This course is not taught in English this year.

1.2. Context and importance of this course in the degree

This course completes the knowledge in the Degree in Computer Engineering related to the organization and architecture of computers in the context of Computer Engineering. The exploitation of parallelism, both vectorial and spatial (multiple processors) is in fact the basic lever of all complex digital systems in society: from cell phones to supercomputers, through all kinds of embedded systems, tablets, laptops, desktop computers or data center servers.

1.3. Recommendations to take this course

To take this course it is recommended to have taken Computer Architecture and Organization 2; Concurrent and Distributed Systems Programming; and Operating Systems.

2. Learning goals

2.1. Competences

Upon passing the course, the student will be more competent to...

- Solve problems and make decisions with initiative, creativity and critical reasoning.
- Use engineering techniques, skills and tools necessary for engineering practice.
- Design and build digital systems, including computers, microprocessor-based systems, and communications systems.
- Develop specific processors and embedded systems, as well as develop and optimize software for such systems.
- Analyze and evaluate computer architectures, including parallel and distributed platforms, as well as develop and optimize software for them.
- Analyze, evaluate and select the most appropriate hardware and software platforms for the support of embedded and real-time applications.

- Analyze, evaluate, select and configure hardware platforms for the development and execution of computer applications and services.

2.2. Learning goals

The student, in order to pass this course, must demonstrate the following results...

- Knows the multiprocessor families, identifies the main components of a multiprocessor and its functions. Understands coherence and consistency problems and their basic solutions. Knows the theory and practice of automatic parallelism extraction.
- Knows the organization of commercial multiprocessors, both those integrated on a chip and those formed by several modules or boards, especially regarding memory and interconnection network.
- Programs simple but multiprocessor-aware algorithms using a multiprocessor programming standard.

2.3. Importance of learning goals

It is clear that from the early stages of software analysis and design it will be necessary to incorporate parallel and/or distributed execution capabilities. For this purpose, it will be very important to know in some detail the possibilities and limitations of the basic technologies.

This course provides an understanding of the current and future landscape of parallel execution information processing platforms. Shared memory multiprocessors are a clear example of this technology. Its market penetration is and will be very large, since practically all commercial chips, for almost all segments, have versions with multiple processors on a single chip. In many cases these chips can be connected together to form large-scale multiprocessors.

Efficient programming, buying or selling multiprocessors, for example, will be of increasing importance in the activity of a computer engineer, and this course will provide the necessary knowledge to better develop these activities.

3. Assessment (1st and 2nd call)

3.1. Assessment tasks (description of tasks, marking system and assessment criteria)

The student must demonstrate that he has achieved the expected learning outcomes through the following assessment activities.

The evaluation will consist of three parts:

- Defense of laboratory practical work (20 points).
- Presentation of results on practical work (20 points).
- Examination of theory and problems (60 points).

To pass the course the student must obtain at least 50 points out of the total and at least 24 points out of 60, that is to say, a 4 out of 10, in the exam. In the case of not achieving a 4 out of 10 in the exam, the student's grade in the call will coincide with the grade obtained in the exam.

The delivery of the results of laboratory practices and practical work will be done coinciding with the dates scheduled for the exam in each call.

4. Methodology, learning tasks, syllabus and resources

4.1. Methodological overview

Follow-up of the learning activities programmed in the course, by means of:

- personalized correction of exercises proposed in class
- tuition
- personalized follow-up in the laboratory session

4.2. Learning tasks

The student will be able to achieve the expected results by doing the following activities:

- Lectures
- Problem-solving classes
- Laboratory practices assistance
- Practical non-presential work
- Personalized tutorials on specific aspects

- Study and personal work

4.3. Syllabus

Module I: Pipelined Vector Processors: Supercomputers

1. Introduction. parallelism
    + Numerical scientific problems
    + Performance of an addition of vectors by scalar processors
        - Pipelined, superpipelined, and superscalar
    + Vector version of the vector addition.
2. Vector Extension of a ld/st architecture
    + Architecture and Organization
    + Basic instruction set (DLXV)
    + Organizations and pipelining
        - Vector register file
        - Functional units (ALUs)
        - Multibank memory (synchronous and concurrent access)
    + Five organizations of vector processor and basic pipelining
    + Performance measures without strip mining: Rn, R?, N?, Nv
    + ZV processor organization: a pipelined vector processor supporting DLXV
3. Two aspects of programming: vector length and vector stride
    + Vector length and strip mining
    + Two schemes for strip mining code generation. AXPY example
    + Performance with strip mining:
        - Assembler example AXPY
        - Rn, R?, N?, Nv when processing noncontiguous elements of a vector (stride)
4. Conflicts in accessing memory banks
    + Introduction. Storage scheme. Fundamental property.
    + Tight Systems
    + Loose Systems
5. DLXV architecture: full instruction set
6. Vector Compilation = automatic extraction of vector operations
    + Introduction
    + Previous transformations that simplify dependency analysis
    + Analysis of dependencies. Dependency graph. Approximate tests
    + Architecture independent optimizations: rename, scalar expansion, vector copy
    + Vectorization
        - Basic Procedure. Full vs. partial vectorization: loop distribution and loop exchange. Reduction
7. Final Thoughts: Amdahl's Law
8. Commercial Vector Processors
    + Introduction
    + Table of Supercomputers
    + Family NEC SX-4 and SX-9 ACE (it may change)
        - Concept of partitioned data path
    + Vector Extensions Intel: from SSE to AVX512 (it may change)

Module II: Shared Memory Multiprocessors

1. Classification of parallel computers from M.J. Flynn
    + SISD, SIMD and MIMD
2. Objectives and problems of the MIMD machines
3. Simple model of H.S. Stone to distribute processes in processors
4. Shared-memory multiprocessors. Overview
    + Architecture-Programming: communication, synchronization, process creation
    + Organization: caches, interconnection network, main memory
5. Interconnection Network

+ Conflict, degradation, topology, cost, circuit switching or packet switching , performance, availability
+ Dynamic Topologies (indirect networks): bus, multibus, crossbar, multi-stage networks
+ Static Topologies (direct network): star, ring, mesh, tree, hypercube

6. Synchronization Mechanisms
+ Instruction set: Test & Set, Fetch & Op, Load Linked
+ Implementation. Combination of requests
+ Barriers

7. Parallel Compilation
+ Automatic extraction of parallel tasks

8. The problem of consistency
+ System, multiprocessor, multi-level cache, more examples
+ Copy-back and write-through

9. The memory model
+ Sequential consistency, pros and cons
+ A definition of consistency

10. coherence protocols based on diffusion
+ Invalidation. Diffusion vs. selective shipping
+ Examples of invalidation + CB + Bus: MSI, EI, Write Once, MESI
+ Snoopy protocols

11. Hierarchy of multilevel caches

12. coherence protocols based on directory
+ Hw requirements and some sample transactions
+ Simple protocol directory

13. Examples of current chip with more than one processor (core)
+ SUN, Intel, AMD, ARM, ...

4.4. Course planning and calendar

Schedule of sessions and labs:

Please see the academic calendar published by EINA.

Expected distribution of student work:

Lectures: 30 hours
Problems: 15 hours
Labs: 15 hours
Personal practice: 12 hours
Personal study: 73 hours
Rating: 5 hours

4.5. Bibliography and recommended resources

http://psfunizar10.unizar.es/br13/egAsignaturas.php?codigo=30237&Identificador=14702