Processor Design
–
Introduction, part I

Professor Jari Nurmi
Institute of Digital and Computer Systems
Tampere University of Technology, Finland
email jari.nurmi@tut.fi

Background

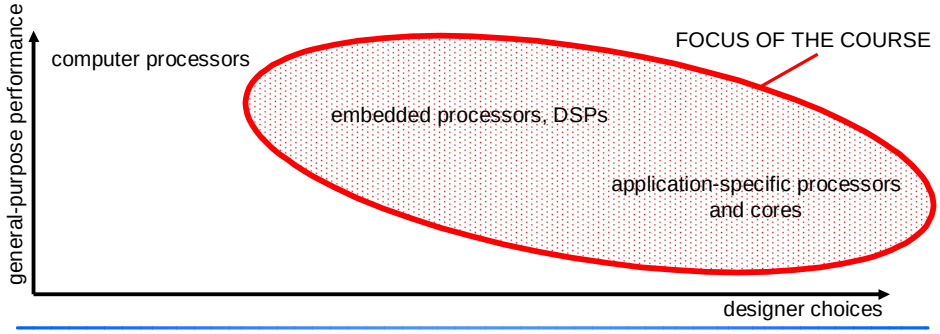
- Some trends in digital system integration
  - System-on-Chip implementation driven by
    - developing circuit technologies (Moore's law)
    - economics of miniatyrization
  - Embedded systems are everywhere
    - cars, elevators, machines, industrial applications
    - household devices, office equipment, games, toys
    - telecommunication equipment, video and audio systems
  - Increasing software content of the integrated systems
    - more functionality integrated on a chip
    - reuse of processors and software modules
    - part of the functionality by programmable processor cores
      - RISC
      - DSP
      - microcontroller
      - VLIW
      - application specific architectures

Consequences of the Trends

- Benefit of the knowledge of processor design aspects in
  - digital design (including application specific architectures!)
  - embedded systems design
  - computer / processor design

Course Target

- The course will deepen the knowledge on
  - processor operating principles
  - design (and choice) of processor architectures
  - processor implementation
  - analysis of performance and other characteristics



Course Outline

- processor basics revisited
- processor design flow
- instruction set design
- requirements set by the operating system
- execution unit, arithmetics
- pipelining and parallelism
- control and interrupts
- memories, exploiting memory hierarchy
- cache and virtual memory management
- input/output system
- examples of processor architectures
- properties of embedded processors
- DSP design specialties
- protocol processors
- benchmarking processor performance and economics

Different Views of a Computer (*

(* according to Heuring – Jordan)

- User's view
  - software
  - speed, storage capacity
  - peripheral device functionality
- Computer architect's view
  - concerned with design & performance
  - optimum programming utility and implementation performance
  - design of hardware to execute instructions
  - use of benchmarking tools and other measures
  - balanced performance of different building blocks
  - performance needs satisfied at lowest cost possible
- Logic designer's view
  - designs the machine at the logic gate level
  - determines whether the architect meets cost and performance goals

Instruction Set Architecture (ISA)

- ISA is the programmer's view of the processor
- Instruction set – the collection of all machine operations (and the related mechanisms to access operands)
- Programmer sees set of instructions, along with the machine resources manipulated by them
- ISA includes
  - instruction set
  - memory
  - programmer-accessible registers of the system

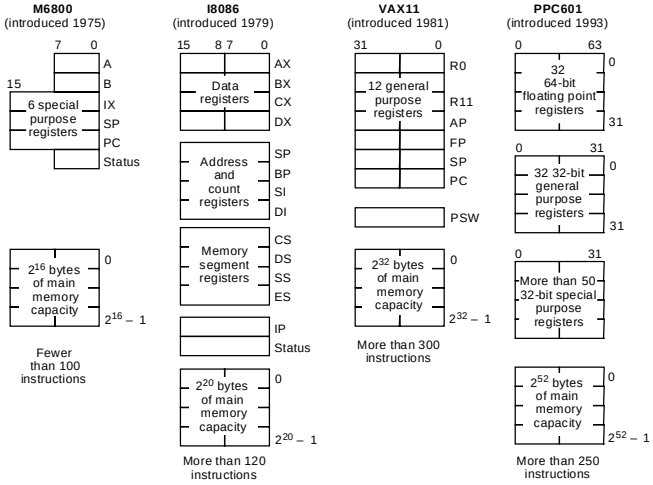
Instruction Set vs. Organization

- The instructions (and the programmer's view of the processor) are the software interface to the processor
- The instruction set may be implemented using different organizational architectures
  - e.g. different number of data buses, execution units, cycles per instruction. . .
  - some organizations are better for the given instruction set, some worse
  - the cost and performance of the organization

Architecture vs. Implementation

- The processor architecture may have several (different) implementations
  - e.g. microprogram or hardwired control
  - different circuit technologies
  - a variation of hidden features (registers etc.)
  - different amounts of pipelining
  - different amounts of parallelism (e.g. superscalar)
- this is a cost and performance issue

Programmer's Models



Source: Heuring – Jordan: Computer Systems Design and Architecture

Architecture Types

- Load-store
  - operations performed to registers
- Register – memory
  - possible to have one source operand in memory
- Register+memory
  - possible to have operands and result destination in memory

- Affects the coding style considerably
- Affects also the achievable clock cycle time and CPI

Dynamic vs. Static Efficiency

- Performance-wise the dynamic efficiency i.e. number of instructions (and the time used) to run a program is important
- Static efficiency
  - number of instructions in a program
  - instruction word length
- The latter counts maybe even more in embedded and integrated systems
- The program size is determined by two variables
  - instruction size (e.g. 16/32 bit)
  - instruction efficiency
    - how much ”work” does a single instruction do
    - RISC/CISC issues etc.

Orthogonality

- Generally, the property of an architecture allowing the use of the same
  - register set
  - data types
  - addressing modes
  - etc.

for all (as many as possible) instruction

- Opposite: specific registers, varying modes
- Orthogonality makes the processor a good target for a compiler (and manual coding!)

The Holy Trinity

- CPI = cycles per instruction
- t_{clk} = clock cycle time
- N = instructions per program

- execution time $T_{ex} = N \times CPI \times t_{clk}$

- Improving performance by
  - decreasing number of instructions
  - decreasing CPI
  - decreasing t_{clk}

Arithmetic – Logic – Control

- Instructions can be categorized to
  - arithmetic instructions
  - logic instructions
  - change-of-flow (control) instructions
  - data transfer

Data Types

- Integers
  - words
  - half-words, bytes
- Fixed-point fractions
  - number of integer and fraction bits
  - 2's complement, sign-and-magnitude, 1's complement
- Floating-point (real) numbers
  - single or double precision
  - different formats
- Decimal digits
- Characters
- Bits (bit vectors)

Fixed-point vs. Floating-point

- Not just a data format . . .
- Precision vs. dynamic range
- Implementation complexity

Branches

- Different types
  - (conditional) branch
  - jump
  - subroutine calls (branch and link)
  - returns
- Denoted branch condition compared to corresponding status bits
- Styles include
  - test...branch instruction pair
  - ”integrated” branch instructions (implicit flags)
  - selective setting of flags in any instruction
  - with and without delay slots

Addressing Modes

- Immediate — *128 0xFFFF 0b1010*
- Absolute (Memory) — *#0x2000 loop_start*
- Register — *Rn*
- Indirect — *[Rn]*
  - register indirect
  - memory indirect (deferred) — *[mem_addr]*
- Indexed — *[Ri + Rj]*
- Displacement (base+offset) — *[Ri + 7]*
- Autoincrement/decrement etc. variations
  - *[Rn]++*
  - *--[Rn]*

End of 1st part of introduction

2nd part will concentrate more on hardware