Micro-programmed Control
Ch 17

Micro-instructions

Micro-programmed
Control Unit

Sequencing

Execution Characteristics

Course Summary

11.10.2002   Copyright Teemu Kerola 2002   1

Hardwired Control (4)

- Complex
- Fast
- Difficult to design
- Difficult to modify
  – lots of optimization work done at
    implementation phase (after design)
  – all optimization work (I.e., most of the work?)
    must be redone after any changes to design

11.10.2002   Copyright Teemu Kerola 2002   2

Micro-programmed Control (3)

- Implement "execution engine" inside CPU
  – execute one micro-instruction at a time
- What to do now?
  – micro-instruction
    - control signals
  – stored in micro-instruction control memory
    - micro-program, firmware
- What to do next?
  – micro-instruction program counter
    - default (?): next micro-instruction
    - jumps or branches?

11.10.2002   Copyright Teemu Kerola 2002   3

Machine Instructions
vs. Micro-instructions

11.10.2002   Copyright Teemu Kerola 2002   4

Machine Instructions
vs. Micro-instructions (2)

- Machine instruction fetch-execute cycle
  produces machine instructions to be
  executed at CPU
- Micro-instruction fetch-execute cycle
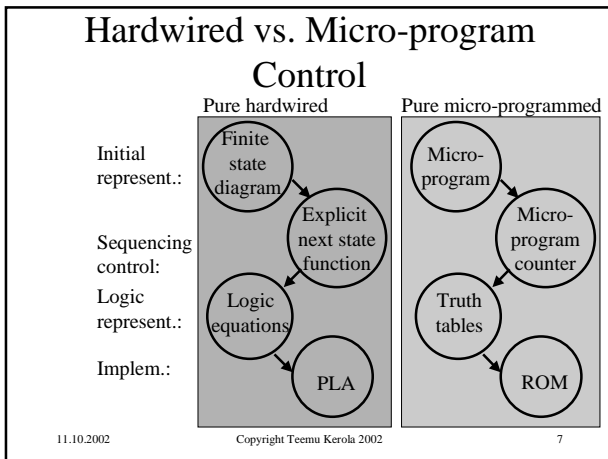 produces control signals for data path

11.10.2002   Copyright Teemu Kerola 2002   5

Micro-program (4)

- Stored in control memory
- ROM, PROM, EPROM
- One "subroutine" for each machine
  instruction
  – one or more micro-instructions
- Defines architecture
  – change instruction set?
    ⇒ reload control memory

Fig. 17.2
(Fig. 15.2 [Stal99])

11.10.2002   Copyright Teemu Kerola 2002   6

Hardwired vs. Micro-program Control

Pure hardwired | Pure micro-programmed

Initial represent.:
Sequencing control:
Logic represent.:
Implem.:

Finite state diagram → Explicit next state function → Logic equations → PLA

Micro-program → Micro-program counter → Truth tables → ROM

11.10.2002   Copyright Teemu Kerola 2002   7

Microcode (3)

- Horizontal micro-code
  (Fig. 15.1 (a) [Stal99])
  Fig. 15.1 (a)
  - control signals directly in micro-code
  - all control signals always there
  - lots of signals ⇒ many bits in micro-instruction
- Vertical micro-code
  (Fig. 15.1 (b) [Stal99])
  Fig. 17.1 (b)
  - each action encoded densely
  - actions need to be decoded to signals at execution time
  - takes less space but may be slower
- Each micro-instruction is <u>also</u> a conditional branch?

11.10.2002   Copyright Teemu Kerola 2002   8

Micro-programmed Control Unit (4)

- Control Address Register
  Fig. 17.4
  (Fig. 15.4 [Stal99])
  - "micro-program PC"
- Control Memory
- Control Buffer Register
  - current micro-instruction
    - control signals
    - next address control
- Sequencing logic
  - select next value for Control Address Reg

11.10.2002   Copyright Teemu Kerola 2002   9

Micro-programming (3)

- Simple design
- Flexible
  - adapt to changes in organization, timing, technology
  - make changes late in design cycle, or even in the field
- Very powerful instruction sets
  - use bigger control memory if needed
  - easy to have complex instruction sets
    - is this good?

11.10.2002   Copyright Teemu Kerola 2002   10

Micro-programming (2)

- Generality
  - multiple instruction sets on same machine
  - tailor instruction set to application?
- Compatibility
  - easy to be backward compatible in one family
  - many organizations, same instruction set

11.10.2002   Copyright Teemu Kerola 2002   11

Micro-programming (3)

- Costly to implement
  - need tools:
    - micro-program development environment
    - micro-program compiler
- Slow
  - micro-instruction interpreted at execution time
  - interpretation is internal to CPU
  - interpret one instruction at a time
- Interpretation control with hardwired logic?

11.10.2002   Copyright Teemu Kerola 2002   12

RISC vs. Micro-programming (8)

- Simple instructions can execute at very high clock rate
- Compilers can produce micro-instructions
  - machine dependent optimization
- Use only simple instructions and addressing mode
- Keep "micro-code" in RAM instead of ROM
- no micro-instruction interpretation logic needed
- Fast access to "micro-code" in RAM via caching
- Skip instruction interpretation of a micro-program and simply compile directly into lowest language of machine?
- ⇒ Compile to "micro-code" and use hardwired control for RISC (e.g., Pentium II)

11.10.2002   Copyright Teemu Kerola 2002   13

Micro-program Sequencing (3)

- Two address format   Fig. 17.6   (Fig. 15.6 [Stal99])
  - most often need next micro-instruction address
    - waste of space to store it most of the time?
  - conditional branch address   (Fig. 15.7 [Stal99])
- One address format   Fig. 17.7
  - (Conditional) branch address
- Variable format
  - only branch micro-instructions have addresses
  - waste of time many times?

11.10.2002   Copyright Teemu Kerola 2002   14

Micro-instruction Explicit Address Generation

- Addresses explicitly present
  - Two-field
    - select one of them
  - Unconditional branch
    - jump to this one
  - Conditional branch
    - select this one or default

11.10.2002   Copyright Teemu Kerola 2002   15

Micro-instruction Implicit Address Generation

- Addresses not explicitly present
  - Mapping
    - map opcode in machine instruction into micro-instruction address   (Fig. 15.9 [Stal99])
  - Addition   Fig. 17.9
    - higher order bits directly from opcode
    - lower order bits based on current status and tag bits, or fields in current microinstruction
  - Residual Control
    - return from micro-program subroutine

11.10.2002   Copyright Teemu Kerola 2002   16

Micro-instruction Encoding

- Usually a compromise between pure horizontal and vertical formats
    Fig. 17.11   (Fig. 15.11 [Stal99])
  - optimize on space with encoding multiple signals into a set of fields
    - each field defines control signals for certain separate actions
    - mutually exclusive actions are encoded into the same field
  - make design simpler by not using maximum encoding

11.10.2002   Copyright Teemu Kerola 2002   17

Micro-instruction Encoding (2)

- Functional encoding
  - each field controls some function
    - load accumulator
    - load ALU operands
    - compute next PC
- Resource encoding
  - each field controls some resource
    - ALU
    - memory

11.10.2002   Copyright Teemu Kerola 2002   18

Different Micro-instruction Sets for a specific "Simple Machine" (3)

(Fig. 15.12 [Stal99])

Fig. 17.12

- Micro-instruction types
  - 3 register transfers, 2 mem ops, 5 ALU ops, 3 seq. ops
- Vertical format

| type | operation | reg |

  - 3 bits for type, 3 bits for operation
  - 2 bits for reg select (max 4 regs)

(Fig. 15.12(a) [Stal99])

Fig. 17.12 (a)

- Horizontal format
  - 2 bits for reg transfers (3 ops + "none")
  - 2 bits for mem ops (2 ops + "none")
  - 2 bits for seq. ops (3 ops + "none")

Fig. 17.12 (b)

(Fig. 15.12(b) [Stal99])

  - 3 bits for ALU ops (5 ops + "none")
  - 2 bits for reg select + 8 bits for constant

11.10.2002   Copyright Teemu Kerola 2002   19

LSI-11 Single Board Processor

11.10.2002   Copyright Teemu Kerola 2002   20

LSI-11 (PDP-11) (5)

- Three-chip single board processor
  - data chip
    - 26 8-bit regs
      - 8 16-bit general purpose regs,
      - PWS, MAR, MBR, ...
    - 8-bit ALU
      - (at least) 2 passes needed for 16-bit reg ops
  - control chip

(Fig. 15.14 [Stal99])

  - control store chip

Fig. 17.14

    - 22 bit wide control mem for micro-instructions
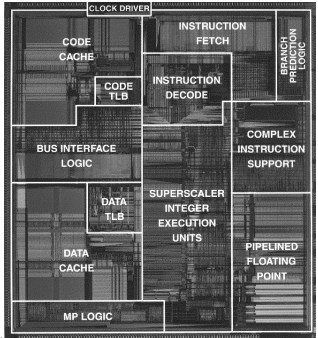 - connected by micro-instruction bus

Fig. 17.13

(Fig. 15.13 [Stal99])

11.10.2002   Copyright Teemu Kerola 2002   21

LSI-11 Micro-instruction Set (2)

- Implements PDP-11 instruction set architecture for LSI-11 hardware
  - e.g., PDP-11 16 bit ALU vs. LSI-11 8-bit ALU
- 22 bit wide, extremely vertical set

Fig. 17.15

  - 4 bits for special functions

(Fig. 15.15 [Stal99])

  - 1 bit for testing interrupts
  - 1 bit for "micro-subroutine return"
  - 16 bits for variable format micro-ops
    - jump, cond. branch, literal ops, reg ops
    - ALU, logical, general, I/O ops

Table 17.5

(Tbl 15.5 [Stal99])

11.10.2002   Copyright Teemu Kerola 2002   22

-- End of Chapter 17 --
-- Micro-programmed Control --

(16.10)

http://infopad.EECS.Berkeley.EDU/CIC/die_photos/pentium.gif

11.10.2002   Copyright Teemu Kerola 2002   23

11.10.2002   Copyright Teemu Kerola 2002   24

Summary (11)

- How clock signals cause instruction executions?
- Low level stuff
  - gates, basic circuits, registers, memory
- Cache
- Virtual memory & TLB
- ALU, Int & FP arithmetic's
- Instruction sets
- CPU structure & pipelining
- Branch prediction, limitations, hazards, issue
- RISC & superscalar processor
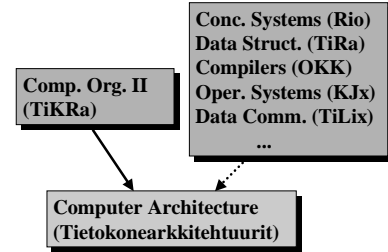- IA-64 & Crusoe
- Hardwired & micro-controlled control

11.10.2002   Copyright Teemu Kerola 2002   25

Want to Know More?

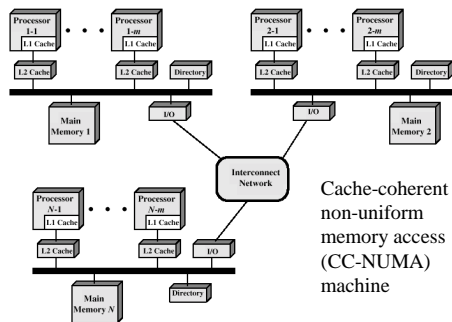- Read the text book completely
- 58070-8 Computer Architecture (4 cr)

Conc. Systems (Rio)
Data Struct. (TiRa)
Compilers (OKK)
Oper. Systems (KJx)
Data Comm. (TiLix)
...

Comp. Org. II
(TiKRa)

Computer Architecture
(Tietokonearkkitehtuurit)

11.10.2002   Copyright Teemu Kerola 2002   26

-- The End of Comp Org II --

Processor 1-1, L1 Cache • • • Processor 1-m, L1 Cache   Processor 2-1, L1 Cache • • • Processor 2-m, L1 Cache

L2 Cache   L2 Cache Directory   L2 Cache   L2 Cache Directory

Main Memory 1   I/O   I/O   Main Memory 2

Interconnect Network

Cache-coherent
non-uniform
memory access
(CC-NUMA)
machine

Processor N-1, L1 Cache • • • Processor N-m, L1 Cache

L2 Cache   L2 Cache I/O

Main Memory N   Directory

(Fig. 18.11)

11.10.2002   Copyright Teemu Kerola 2002   27