



### Administrative Stuff

---


- Homework1 due NOW
  - No late homework accepted
- Homework2 available on the www
  - Due Thursday 10/10
- Make sure you
  - Sign up for mailing list [cs122a@lists.cs.ucr.edu](mailto:cs122a@lists.cs.ucr.edu)
    - Only 19 out of 30 students signed up so far!!!
  - Sign up for presentation
    - Only 27 out of 30 students signed up so far!!!
- CS193 credits available
  - Project on design and compile (gcc) embedded system benchmarks (EMBCC) to reconfigurable platform (RT-RISC)




CS122A: Embedded System Design, Fall02
1

## General-Purpose Processors


### Software


CS122A: Embedded System Design, Fall02
2

### Outline

---


- Basic architecture
- Operation
- Programmer's view
- Development environment
- ASIPs
- Selecting a microprocessor
- General-purpose processor design


CS122A: Embedded System Design, Fall02
3

### General Purpose Processor

---


- Processor designed for a variety of computation tasks
- Advantage to processor manufacturer
  - Low unit cost
    - Manufacturer spreads NRE over large numbers of units
    - Motorola sold in excess of 500,000,000 68HC05 microcontrollers per year!
    - Carefully designed since higher NRE is acceptable
    - Can yield good performance, size, and power
- Advantage to embedded system designer
  - User just writes software; no processor design
    - Low NRE cost
    - Short time-to-market/prototype
    - High flexibility



CS122A: Embedded System Design, Fall02
4

### 68HC05 in automobiles

---

- Electric seat control (& storage)
- Audio system (& storage)
- Cruise control
- Ignition system
- Car alarm
- Power window
- Keyless entry
- Airbags






CS122A: Embedded System Design, Fall02
5

### 68HC05 in computer peripherals

---

- Monitor
- Keyboard control
- Cordless mice
- Trackballs
- Gamepad




CS122A: Embedded System Design, Fall02
6

### 68HC05 in consumer electronics

- CD players
- Remote control
- Washing machines
- Oven control
- LCD display
- Watches




CS122A: Embedded System Design, Fall02

7

### 68HC05 in industrial application

- Programmable logic controller
- Data acquisition system
- Smoke detectors
- Security devices
- Thermostats
- Furnace ignition systems
- Factory automation
- Sensor
- Switch applications




CS122A: Embedded System Design, Fall02

8

### 68HC05 in telecommunication

- Pagers
- Answering machine
- Telephone control
  - Auto-dialing
  - Number storage
  - Display control

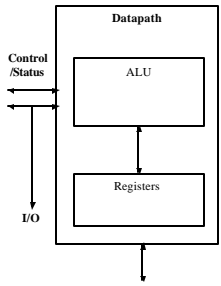


CS122A: Embedded System Design, Fall02

9

### Datapath

- ALU
  - +-&|'>...
  - =0, c,
- N-bit processor
  - N-bit
    - ALU
    - registers
    - buses
    - memory data interface
  - N = 8, 16, 32, or 64
    - word
  - External bus may be narrower

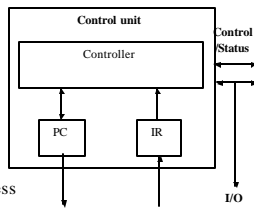


CS122A: Embedded System Design, Fall02

10

### Control unit

- Performs
  - Instruction Fetch
  - Dataflow control
    - ALU operation
    - Memory operation
    - Register operation
    - PC control
- PC bitwidth
  - Determine the direct address space

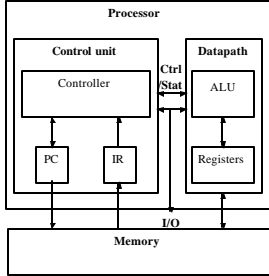


CS122A: Embedded System Design, Fall02

11

### Basic Architecture

- Typical sequence
  - Instruction fetch
  - Instruction decode
  - Operand fetch
  - Execute
  - Store result
- Critical path through DP
  - Clock cycle
  - Impact performance
- RISC architecture
  - Data dependent pipeline
    - Difficult to predict timing



CS122A: Embedded System Design, Fall02

12

### Two Memory Architectures

- Princeton
  - Program and data words share same memory space
  - Fewer memory wires
- Harvard
  - Distinct memory spaces
  - Simultaneous program and data memory access
  - Intel 8051
  - Program may be implemented with ROM

CS122A: Embedded System Design, Fall02 13

### \$ Memory

- Speed up memory access time
- Cache is small but fast memory close to processor
  - Holds a copy of part of the memory
    - Neighboring memory locations recently accessed
  - Hits and Misses
    - Hit to miss ratio must be high to be effective
- Temporal locality
  - E.g. small computation core
- Spatial Locality
  - E.g. sequential program, array

CS122A: Embedded System Design, Fall02 14

### Outline

- Basic architecture
- Operation
- Programmer's view
- Development environment
- ASIPs
- Selecting a microprocessor
- General-purpose processor design

CS122A: Embedded System Design, Fall02 15

### Pipelining: increasing instruction throughput

CS122A: Embedded System Design, Fall02 16

### Pipelining: increasing instruction throughput

- Divide instruction into equal stages
- Instruction must take same number of time
- Pipeline hazard
  - Structure: no enough processing unit
  - Control: branch (loop, if-then-else)
  - Data: RAW, WAR, WAW
- Pipeline scheduling

CS122A: Embedded System Design, Fall02 17

### Structure hazard

- Resource conflict when HW can't support overlap
  - E.g. Functional unit not pipelined
    - Single memory access in Princeton architecture
- Must stall or reschedule instructions

CS122A: Embedded System Design, Fall02 18

### Control and data hazard

- Control hazard
  - A.k.a. branch hazard
  - Avoid stall by
    - Branch prediction:
    - ...loop branch taken at ~90%
- Data Hazard
  - Read After Write (RAW)
    - a=b+c; d=a+b
  - Write After Read (WAR)
    - Possible only if some instruction write early, like autoincrement
    - a=b+c; c=b++
  - Write After Write (WAW)
    - Possible only if some instruction write early
    - b=a+c; d=b; b=0
  - Read After Read (RAR)
    - Is fine

pipelined instruction execution

19  
CS122A: Embedded System Design, Fall02

### Superscalar and VLIW

- Superscalar
  - Execute two or more scalar operation in parallel
    - Scalar: not vector or matrix
  - May have to schedule instruction dynamically
- Very Long Instruction Word
  - Statically encode many scalar operations in one instruction
  - Less run time overhead, more compile time decision
- Is it possible to be superscalar but not VLIW? **YES**
- Is it possible to be VLIW but not superscalar? **NO**

20  
CS122A: Embedded System Design, Fall02

### Outline

- Basic architecture
- Operation
- Programmer's view
- Development environment
- ASIPs
- Selecting a microprocessor
- General-purpose processor design

21  
CS122A: Embedded System Design, Fall02

### Programmer's view

- Architectural abstraction
  - Programmer need not know details of processor's architecture or operation
- Different levels of abstraction, from highest to lowest
  - Structured-language programming
    - for (int I=10; i!=0; i--) total +=i;
    - Processor independent
  - Assembly-language programming
    - MOV, JZ, R1, R2...
    - Processor dependent
  - Machine-language programming
    - 0's and 1's
- The tougher the application, the lower the designer goes
  - Most embedded programmer
    - Work at structured-language level
    - With some assembly level especially for driving external devices

22  
CS122A: Embedded System Design, Fall02

### Assembly-language programming

- Instruction set
  - Allowable atomic operation
  - Bit configuration allowed by IR
- Instructions consist of
  - Opcode field
  - Operand fields
- Types of instructions
  - Data-transfer
    - Move data b/t register, memory, I/O
  - Arithmetic/logical
    - ALU instructions
    - Instruction to move data in/out of ALU
  - Branches
    - Conditional/unconditional jump
    - Call/return (implement procedure/function)

Instructions stored in memory

Instruction 1	opcode	operand1	operand2
Instruction 2	opcode	operand1	operand2
Instruction 3	opcode	operand1	operand2
Instruction 4	opcode	operand1	operand2
...			

23  
CS122A: Embedded System Design, Fall02

### Operand field addressing modes

Addressing mode	Operand field	Register-file contents	Memory contents
Immediate	Data	Data	Data
Register-direct	Register address	Data	Data
Register indirect	Register address	Memory address	Data
Direct	Memory address	Data	Data
Indirect	Memory address	Memory address	Data

Direct: Variables  
Indirect: Pointers

24  
CS122A: Embedded System Design, Fall02

### A few more addressing modes

- Inherent, or implicit
  - Particular reg or mem is in the opcode
    - JAZ: jump on accumulator zero
- Index
  - Direct or indirect add to index register
- Relative
  - Relative to current address

25

CS122A: Embedded System Design, Fall02

### A simple 7-instruction set

Assembly instruct.	First byte	Second byte	Operation
MOV Rn, direct	0000 Rn	direct	Rn = M(direct)
MOV direct, Rn	0001 Rn	direct	M(direct) = Rn
MOV @Rn, Rm	0010 Rn	Rm	M(Rn) = Rm
MOV Rn, #immed.	0011 Rn	immediate	Rn = immediate
ADD Rn, Rm	0100 Rn	Rm	Rn = Rn + Rm
SUB Rn, Rm	0101 Rn	Rm	Rn = Rn - Rm
JZ Rn, relative	0110 Rn	relative	PC = PC + relative (only if Rn is 0) Use assembler to compute "relative" from labels

26

CS122A: Embedded System Design, Fall02

### Sample Programs

C program	Equivalent assembly program
	0 MOV R0, #0; // total = 0
	1 MOV R1, #10; // i = 10
	2 MOV R2, #1; // constant 1
	3 MOV R3, #0; // constant 0
int total = 0;	Loop: JZ R1, Next; // Done if i=0
for (int i=10; i!=0; i--)	5 ADD R0, R1; // total += i
total += i;	6 SUB R1, R2; // i--
// next instructions...	7 JZ R3, Loop; // Jump always
	Next: // next instructions...

27

CS122A: Embedded System Design, Fall02

### Embedded system programming awareness

- Available memory for program and data
  - fit in on-chip memory if possible
- Number and types of registers
  - For general-purpose data storage
  - For special-functions like writing and reading external pins
  - For programming peripherals
- Available I/O facilities for communication
- Interrupts
  - For quick, infrequent, unpredictable tasks
  - Jump to an Interrupt Service Routine (ISR)
  - Return to interrupted point when done
  - Less hardware overhead than Direct Memory Access (DMA)
  - Sample applications
    - Pressing of a button, updating of a status...

28

CS122A: Embedded System Design, Fall02

### Outline

- Basic architecture
- Operation
- Programmer's view
- Development environment
- ASIPs
- Selecting a microprocessor
- General-purpose processor design

29

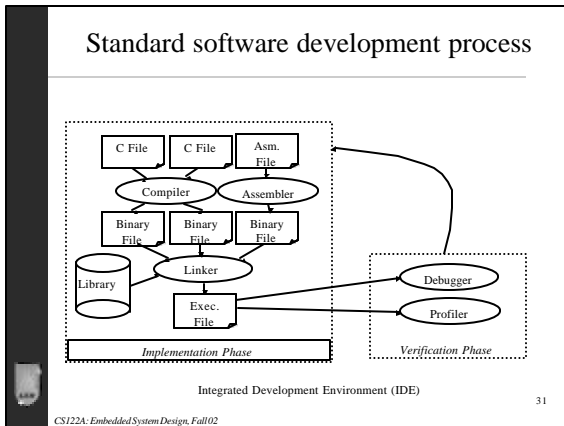
CS122A: Embedded System Design, Fall02

### A tale of two processors

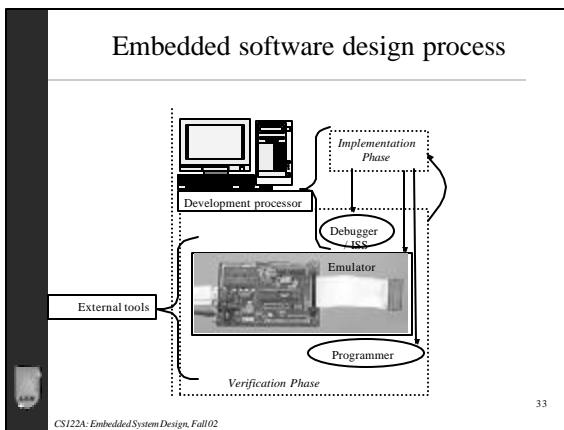
- Development processor
  - Write and debug program
  - Pentium, Sparc...
- Target processor
  - Load program and become part of the embedded system
  - 68HC11, 8051...

30

CS122A: Embedded System Design, Fall02



- ### Implementation phase
- Assembler
    - Translate assembly instructions to machine instructions
    - Opcode and operand into binary counterpart
    - Labels into address offset
  - Compiler
    - Translate structured programs to machine programs
      - With lots of optimization for size/performance
    - Cross compiler
      - Executes on development processor but generates code for target processor
  - Linker
    - Combine separately assembled or compiled files into single program
      - May link in library files
- CS122A: Embedded System Design, Fall02 32



- ### Verification phase
- Typically more difficult for embedded systems
    - Timeliness as well as correct results
      - Real-time: must compute results within timing constraints
    - Coordination with other components of system
  - Debuggers
    - Instruction-set simulator (ISS), virtual machine (VM)
    - Fast time-to-verify but inaccurate
      - Run on development processor
      - "Simulated" environment
  - Emulators
    - Slower time-to-verify but more accurate
      - Run on "target" processor, rig up for debug reason
      - Run in real environment
  - Device programmers
    - Highest accuracy but slowest time-to-verify, least debug control
- CS122A: Embedded System Design, Fall02 34

### Instruction-set simulator (ISS) for a simple processor

• ISS: runs on one processor and executes instructions of another processor

```

typedef struct {
    unsigned char first_byte, second_byte;
} instruction;

instruction program[1024]; //instruction memory
unsigned char memory[256]; //data memory

void run_program(int num_bytes) {

    int pc = -1;
    unsigned char reg[16], fb, sb;

    while( ++pc < (num_bytes / 2) ) {
        fb = program[pc].first_byte;
        sb = program[pc].second_byte;
        switch( fb >> 4 ) {
            case 0: reg[fb & 0x0f] = memory[sb]; break;
            case 1: memory[sb] = reg[fb & 0x0f]; break;
            case 2: memory[(reg[fb & 0x0f]]] = reg[sb >> 4]; break;
            case 3: reg[fb & 0x0f] = sb; break;
        }
    }
}
    
```

MOV Rn, direct  
 MOV direct, Rn  
 MOV @Rn, Rn  
 MOV Rn, #immed.  
 ADD Rn, Rn  
 SUB Rn, Rn  
 JZ Rn, relative

CS122A: Embedded System Design, Fall02 35

- ### Outline
- Basic architecture
  - Operation
  - Programmer's view
  - Development environment
  - ASIPs
  - Selecting a microprocessor
  - General-purpose processor design
- CS122A: Embedded System Design, Fall02 36

### Application-specific Instruction-set processors (ASIPs)


- Specific to an application or application domain
  - Control
  - Data processing
  - Networking
- Spread NRE cost over many applications within the domain
  - Processor design, compiler, linker, etc.
- Programmed by writing software
  - Shorter time-to-market, better flexibility

37

CS122A: Embedded System Design, Fall02

### Microcontroller

- Cheap, small device contains
  - Processor core
  - Timers, A2D, Serial communication, ROM, RAM
  - A.k.a. embedded processor (when they get big)

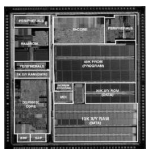


38

CS122A: Embedded System Design, Fall02

### Digital Signal Processor

- Usually more expensive than microcontroller
  - Higher performance for signal processor
  - Large register files, memory blocks, multiplier...
  - Special MAC units and MAC instruction
    - For use in filtering and matrix transformation
  - May contain A2D, D2A, PWM, DMA, timer, or counter




39

CS122A: Embedded System Design, Fall02

### Motorola DSP56301

- 80/100 MIPS with a 80/100 MHz clock at 3.3 V
- Fully pipelined 24 x 24-bit parallel MAC
- 56-bit parallel barrel shifter
- 24-bit or 16-bit arithmetic support under software control
- Addressing modes optimized for DSP applications
- Nested hardware DO loops
- Fast auto-return interrupts
- On-chip concurrent six-channel DMA controller
- On-chip Phase Lock Loop (PLL) and clock generator
- 1024-4096 x 24-bit Program RAM
- 2048/3072 x 24-bit data RAM
- 32-bit parallel PCI/Universal Host Interface (HI32).
- Two Enhanced Synchronous Serial Interfaces (ESSI)
- Serial Communications Interface (SCI) with baud rate generator
- Triple timer module
- 42 programmable General Purpose I/O pins GPIO



40

CS122A: Embedded System Design, Fall02

### Network processor

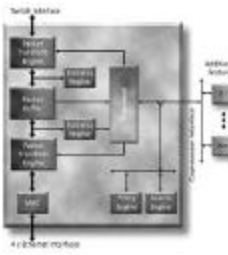
- Packets are 1<sup>st</sup> class citizen
  - QoS guarantee
  - Secure transmission
  - Intelligent/dynamic routing and switching
- Very limited programmability
- Very high network performance
- Low NRE cost
- Moderate flexibility for network application
  - Each switch maker can program to meet
    - Local traffic requirement
    - Local policy requirement
- Operate at up to 10Gbit/S

41

CS122A: Embedded System Design, Fall02

### MMC's EPIF-200 Packet Processor

- Multi-tasking processor
- Programmable modules
  - Packet transform
  - Statistics engine
  - Policy engine
    - Packet classification
    - Future-proof
  - Search engine
    - Longest prefix match routing
- Fast ethernet interfaces



42

CS122A: Embedded System Design, Fall02

### Selecting a microprocessor

- Non-technical aspect
  - Prior expertise
  - Development environment
  - Special licensing
- Technical Aspect:
  - Desired power, cost, speed, performance
  - Clock speed may not be meaningful
    - Number instruction per cycle may differ
  - Instruction per second may not be meaningful
    - Complexity of instruction may differ
  - Dhrystone benchmark, MIPS, EEMBC

43

CS122A: Embedded System Design, Fall02

### Dhrystone & MIPS

- Dhrystone benchmark
  - Artificial benchmark exercise
    - Integer arithmetic
    - String manipulation
  - MIPS
    - 1757 Dhrystones/Second
    - VAX 11/780 is a 1 MIPS machine
- Whetstone benchmark
  - Mainly use for scientific computation
    - Integer
    - Boolean
    - Floating-point
    - Subroutine call
    - Transcendental functions

44

CS122A: Embedded System Design, Fall02

### EEMBC Benchmark

- EDN Embedded Benchmark Consortium
  - Automotive/industrial
    - PWM, Matrix arithmetic, FFT, DCT, IIR filter, FIR filter, Floating point,
  - Consumer electronics
    - JPEG Com/decom, filter, RGB conversion
  - Networking
    - Packet flow, Route lookup, Dijkstra
  - Office automation
    - Text processing, image rotation, dithering, Bezier curve
  - Telecommunications
    - GSM decoder, FFT, Bit allocation, Convolutional Encoder, Auto-Correlation
- All measured for size and speed

45

CS122A: Embedded System Design, Fall02

### Automotive/Industrial Benchmarks

Processor	Approx. Automark Score
STMicro ST20C2 50 MHz	10
NEC VR3000 - 250 MHz	15
Motorola MPC333 - 40MHz	10
Mitsubishi M16C80 - 20 MHz	10
Infineon Triosne TC100P - 80MHz	10
EM PowerPC 750CF - 500 MHz	10
AMD K6-III+ 330MHz/4CR	10

46

Copyright © 1998-2001 Embedded Microprocessor Benchmark Consortium (EEMBC), Inc.  
CS122A: Embedded System Design, Fall02

### Consumer Benchmarks

Processor	Approx. Consumermark Score
TM1200 - 166 MHz	77
ST20C2 50 MHz	10
NEC VR3000 - 250 MHz	15
National Geode GNTI - 200	10
AMD K6-III+ 330MHz/4CR	10

47

Copyright © 1998-2001 Embedded Microprocessor Benchmark Consortium (EEMBC), Inc.  
CS122A: Embedded System Design, Fall02

### Networking Benchmarks

Processor	Approx. Netmark Score
Thyrite TMPR327F - 133MHz	2.5
NEC VR3000 - 250 MHz	4.5
ID775R/CM373-250MHz	5.5
AMD K6-III+ 330MHz/4CR	7.5

48

Copyright © 1998-2001 Embedded Microprocessor Benchmark Consortium (EEMBC), Inc.  
CS122A: Embedded System Design, Fall02

### General-purpose processor design

- Similar design technique as used for single-purpose processor
- Key differences:
  - Program stored in external memory, can be changed
  - Datapath has no knowledge of program when designed

General-purpose      Single-purpose

CS122A: Embedded System Design, Fall 02

### A Simple Microprocessor

Declarations:

```

bit PC[16], IR[16];
bit M[64], RF[16][16];
    
```

MOV Rn, direct  
 MOV direct, Rn  
 MOV @Rn, Rm  
 MOV Rn, #immed.  
 ADD Rn, Rm  
 SUB Rn, Rm  
 JZ Rn, relative

Aliases:

```

op IR[15..12]   dir IR[7..0]
m  IR[11..8]   imm IR[7..0]
r  IR[7..4]    rel IR[7..0]
    
```

50

CS122A: Embedded System Design, Fall 02

### A Simple Microprocessor

Reset PC=0;  
 Fetch IR=M[PC];  
 PC=PC+1  
 from states below

Decode

- Mov1 RF[m] = M[dir] to Fetch
- op = 0000
- Mov2 M[dir] = RF[m] to Fetch
- 0001
- Mov3 M[m] = RF[m] to Fetch
- 0010
- Mov4 RF[m] = imm to Fetch
- 0011
- Add RF[m] = RF[m] + RF[r] to Fetch
- 0100
- Sub RF[m] = RF[m] - RF[r] to Fetch
- 0101
- Jz PC = (RF[m] = 0) ? rel : PC to Fetch
- 0110

MOV Rn, direct  
 MOV direct, Rn  
 MOV @Rn, Rm  
 MOV Rn, #immed.  
 ADD Rn, Rm  
 SUB Rn, Rm  
 JZ Rn, relative

51

CS122A: Embedded System Design, Fall 02

### Architecture Of A Simple Microprocessor

52

CS122A: Embedded System Design, Fall 02

### Architecture Of A Simple Microprocessor

53

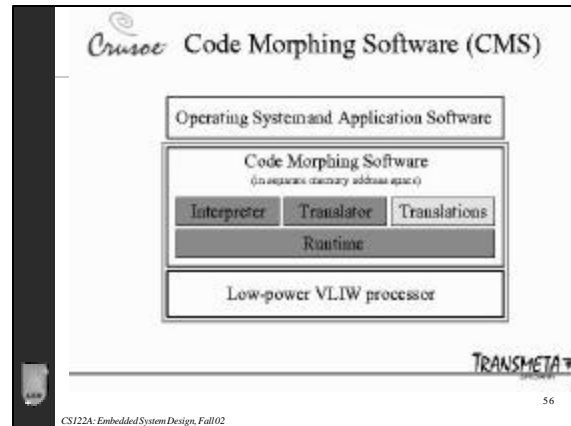
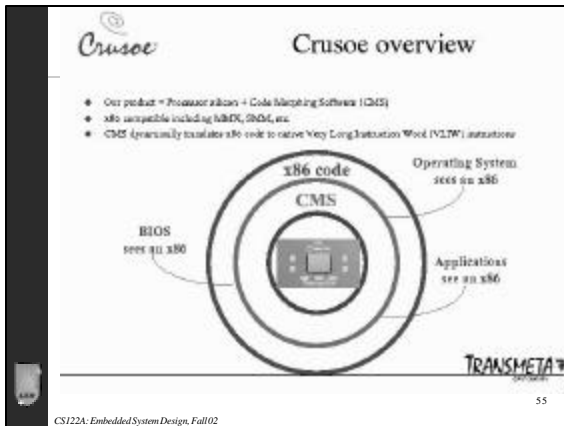
CS122A: Embedded System Design, Fall 02

### Architecture Of A Simple Microprocessor

- storage devices for each declared variable
  - register file holds each of the variables
- functional units to carry out the FSM operations
  - an ALU is used to carry out the required operations
- connections added among the components' ports corresponding to the operations required by the FSM
- unique identifiers created for every control signal

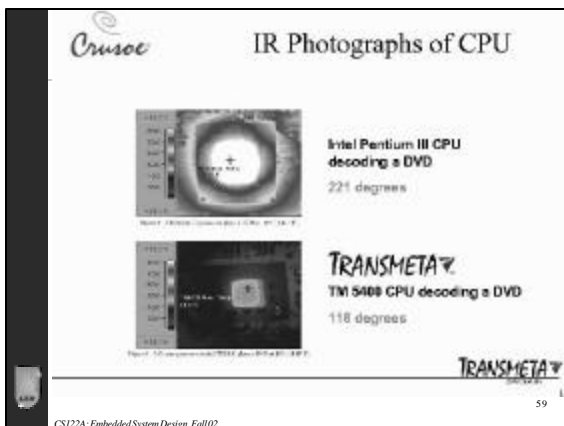
54

CS122A: Embedded System Design, Fall 02



- Crusoe's CMS**
- Interpreter
    - Interpret one x86 instruction at a time
      - Low overhead, but slow running code
  - Translator
    - Translate frequently executed x86 code into native code
    - Translate multiple instruction into one VLIW instruction
      - High overhead, but fast running code
  - Translation cache
    - Optimized for common case
  - Runtime
    - Manage translation cache
    - Quick lookup
    - Chain translations for efficiency
- TRANSMETA
- 57
- CS122A: Embedded System Design, Fall02

- The skinny on Crusoe**
- Allow infrequent-used logic to be in software
    - Expensive logic vs. cheap memory
  - More flexible
    - Easy upgrade and work around silicon bugs
  - Can improve independently and concurrently
    - Process technology
    - Circuit
    - Architecture
    - Software
  - Bottom line
    - Perform 70-110% of comparable chip (pentium III) at 30% of the power
    - Well, 30% of power of the processor
      - Screen, hard-drive are actually bigger power hog
      - At least the heatsinks are smaller...
- TRANSMETA
- 58
- CS122A: Embedded System Design, Fall02



- Administrative Stuff**
- Homework1 due 80 minutes ago
    - No late homework accepted
  - Homework2 available on the www
    - Due Thursday 10/10
  - Make sure you
    - Sign up for mailing list [cs122a@lists.cs.ucr.edu](mailto:cs122a@lists.cs.ucr.edu)
      - Only 19 out of 30 students signed up so far!!!
    - Sign up for presentation
      - Only 27 out of 30 students signed up so far!!!
  - CS193 credits available
    - Project on design and compile (gcc) embedded system benchmarks (EMBCC) to reconfigurable platform (RT-RISC)
- TRANSMETA
- 60
- CS122A: Embedded System Design, Fall02