Processors

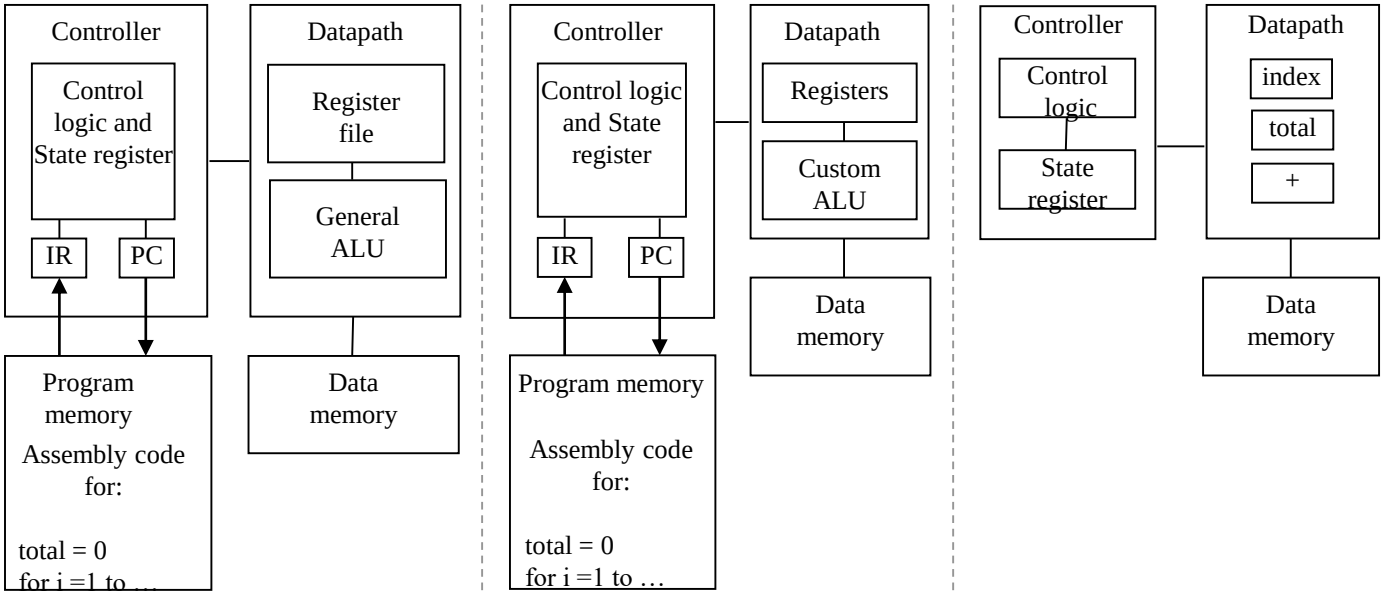
Shao-Yi Chien

Outline

- Processor technology
- Basic architecture
- Operation
- Programmer's view
- Developed environment
- CPU power consumption
- Application-specific instruction-set processors (ASIP)
- Co-processor
- Selecting a microprocessor
- Other trends of processor design

Processor Technology

- The architecture of the computation engine used to implement a system's desired functionality
- Processor does not have to be programmable
  - "Processor" *not* equal to general-purpose processor



General-purpose ("software") **Application-specific** **Single-purpose** ("hardware")

Processor Technology

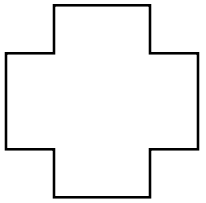
- Processors vary in their customization for the problem at hand

Desired functionality

```
total = 0
for i = 1 to N  loop
   total += M[i]
end loop
```

General-purpose processor

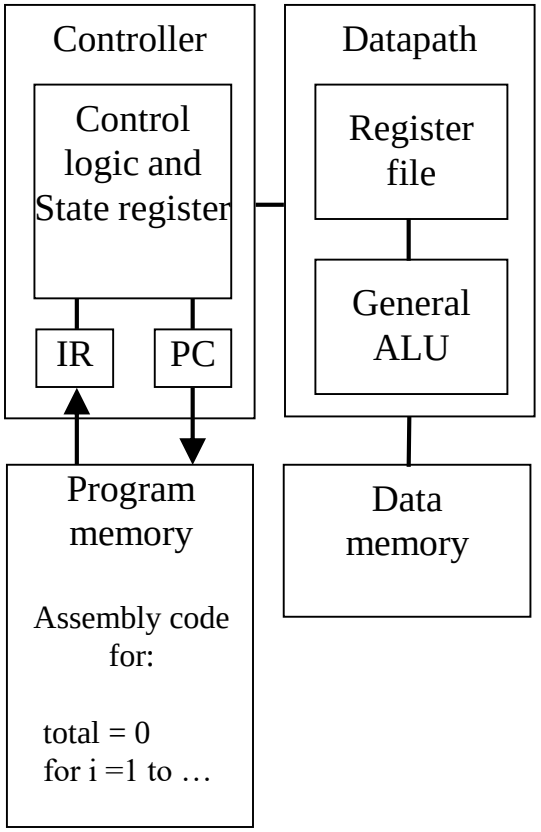
Application-specific processor



Single-purpose processor

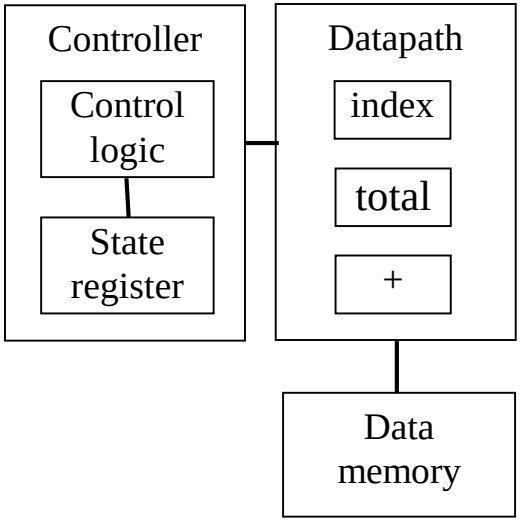
General-Purpose Processors

- Programmable device used in a variety of applications
  - Also known as “microprocessor”
- Features
  - Program memory
  - General datapath with large register file and general ALU
- User benefits
  - Low time-to-market and NRE costs
  - High flexibility
- Intel CPU is the most well-known, but there are hundreds of others



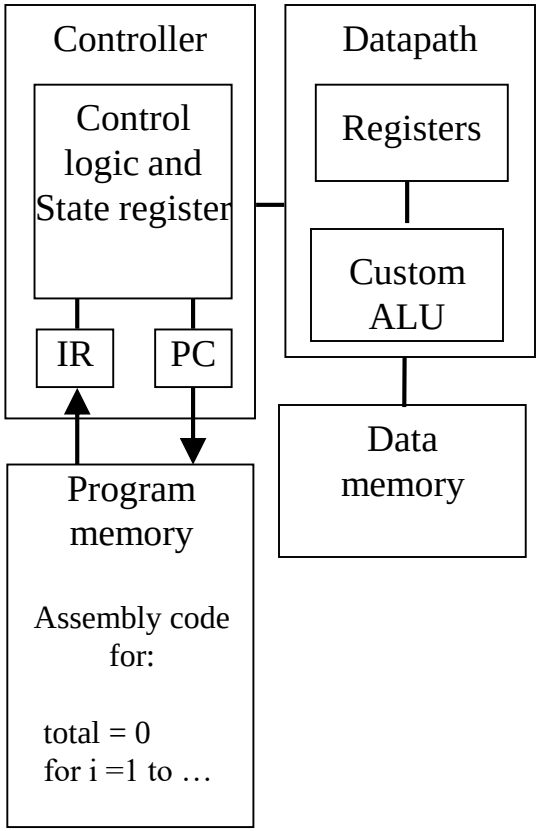
Single-Purpose Processors

- Digital circuit designed to execute exactly one program
  - a.k.a. coprocessor, accelerator or peripheral
- Features
  - Contains only the components needed to execute a single program
  - No program memory
- Benefits
  - Fast
  - Low power
  - Small size



Application-Specific Processors

- Programmable processor optimized for a particular class of applications having common characteristics
  - Compromise between general-purpose and single-purpose processors
- Features
  - Program memory
  - Optimized datapath
  - Special functional units
- Benefits
  - Some flexibility, good performance, size and power

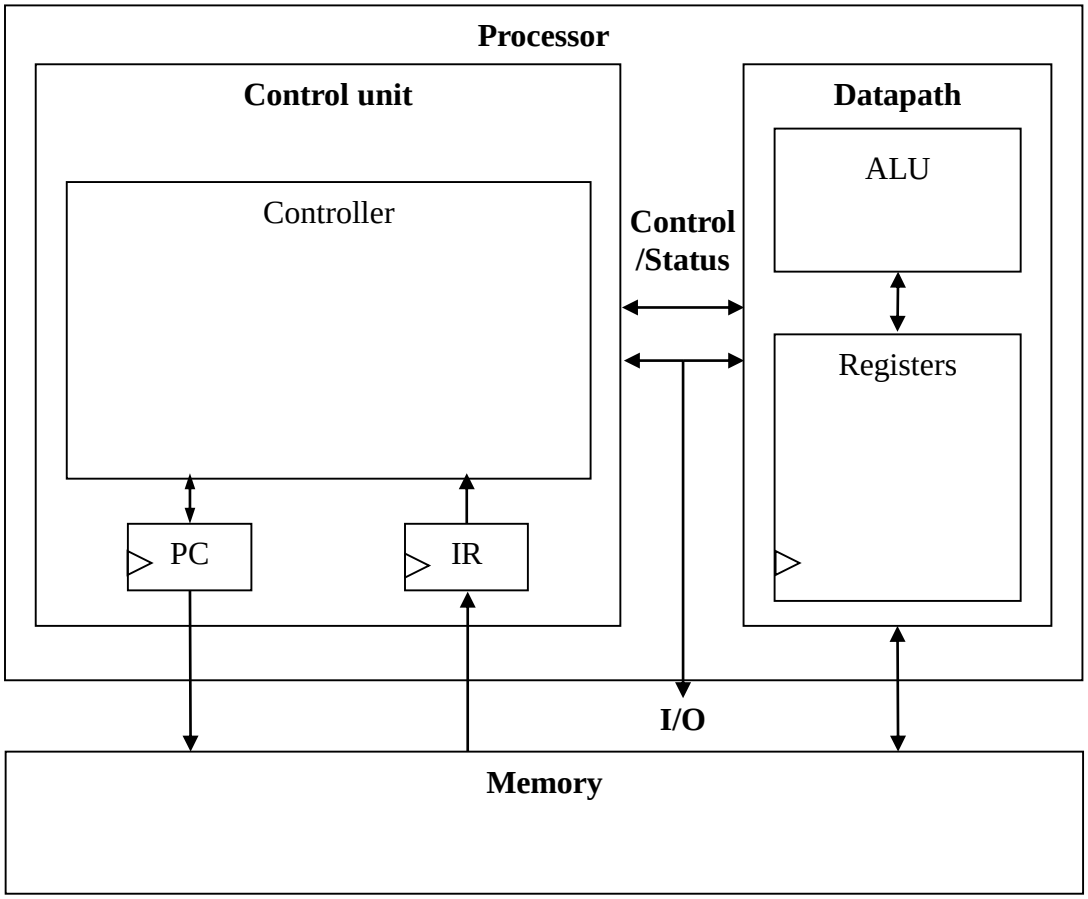


Why General-Purpose Processors in SoCs?

- Using microprocessors is a very efficient way to implement digital systems
- Microprocessors make it easier to **design families of products** that can be built to provide various feature sets at different price points and **can be extended to provide new features** to keep up with rapidly changing markets

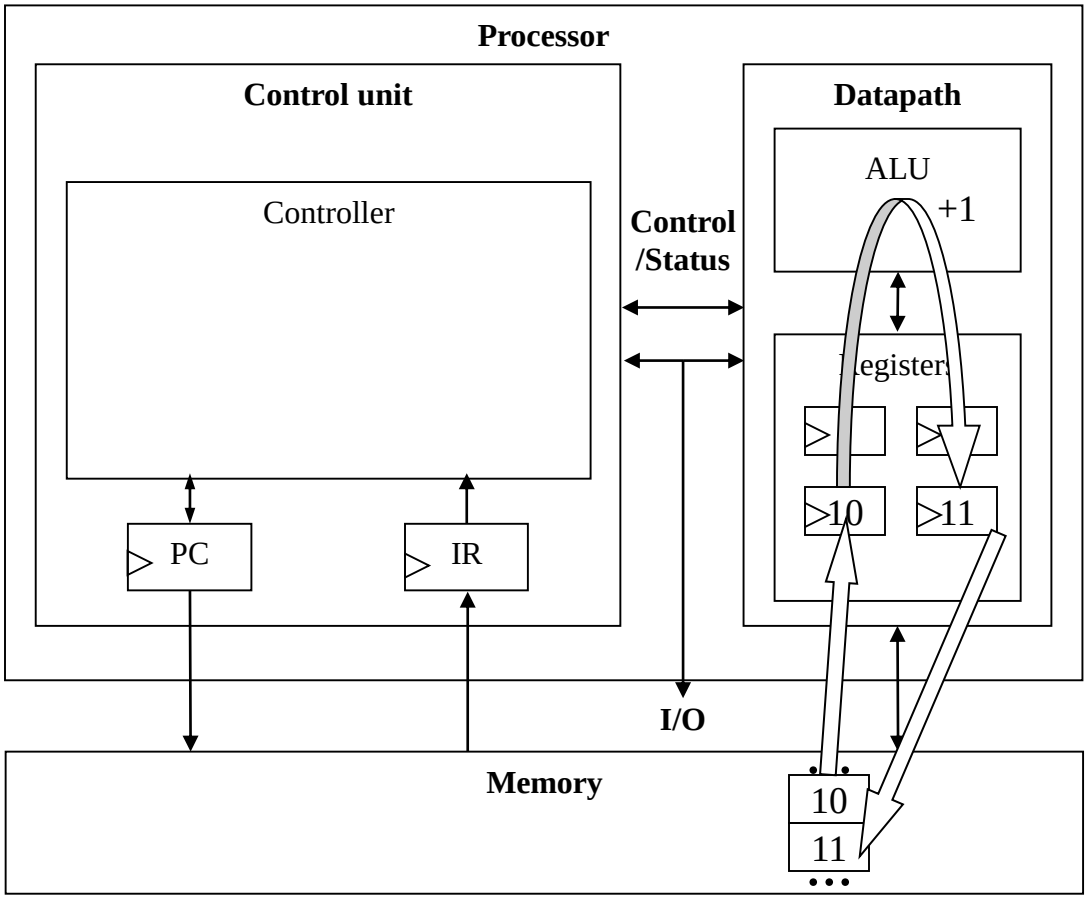
Basic Architecture

- Control unit and datapath
- Key differences to single-purpose processors
  - Datapath is general
  - Control unit doesn't store the algorithm – the algorithm is "programmed" into the memory



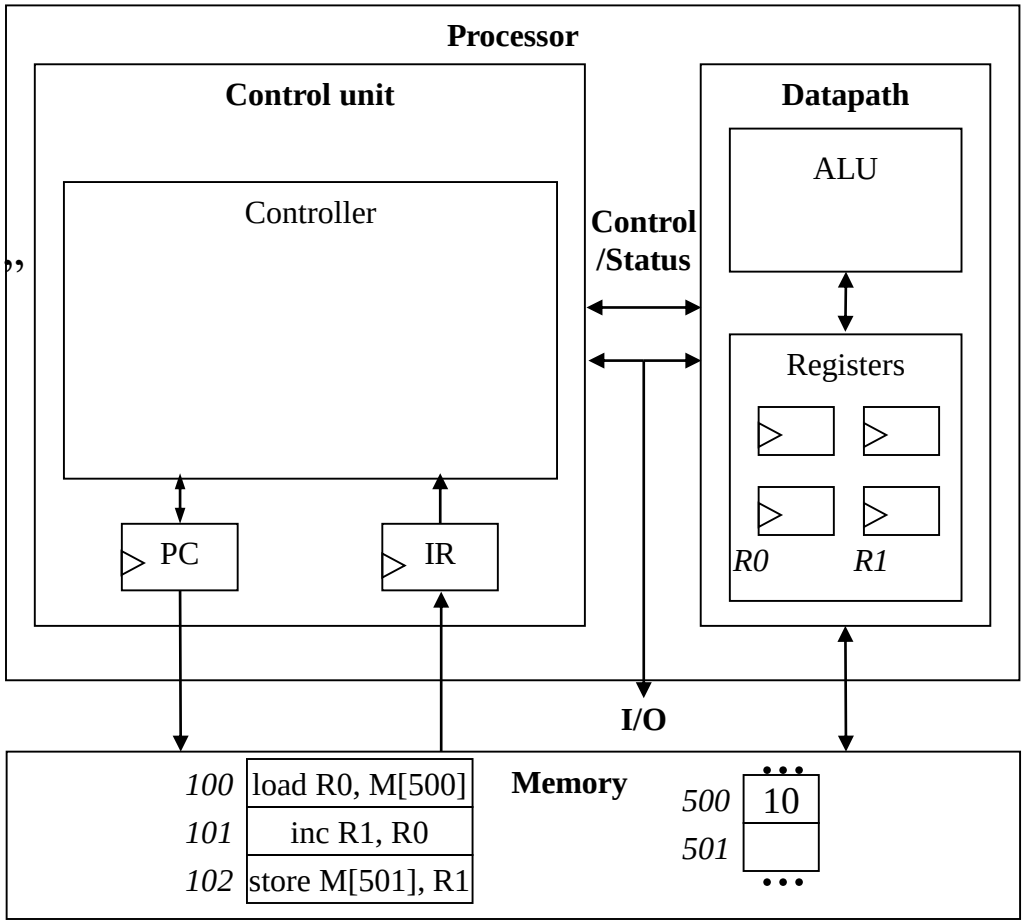
Datapath Operations

- Load
  - Read memory location into register
- ALU operation
  - Input certain registers through ALU, store back in register
- Store
  - Write register to memory location



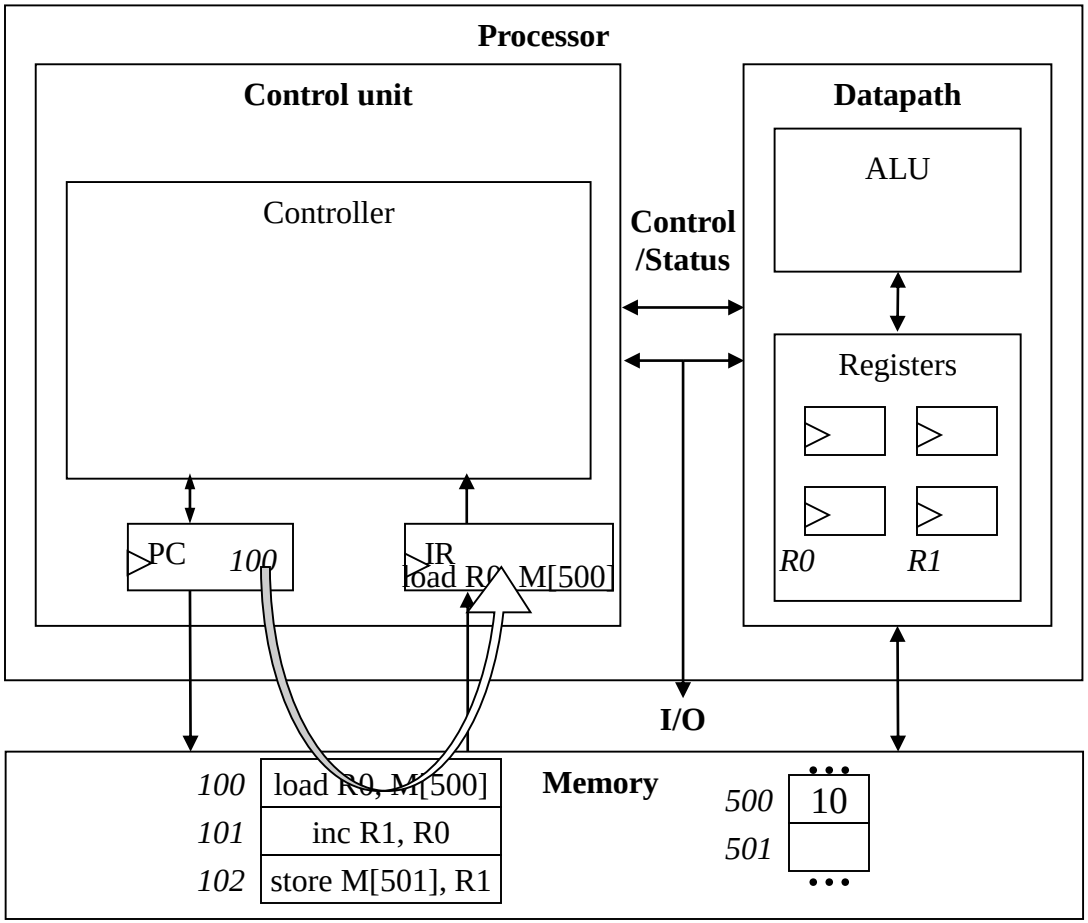
Control Unit

- Control unit: configures the datapath operations
  - Sequence of desired operations (“instructions”) stored in memory – “program”
- Instruction cycle – broken into several sub-operations, each one clock cycle, e.g.:
  - Fetch
  - Decode
  - Fetch operands
  - Execute
  - Store results



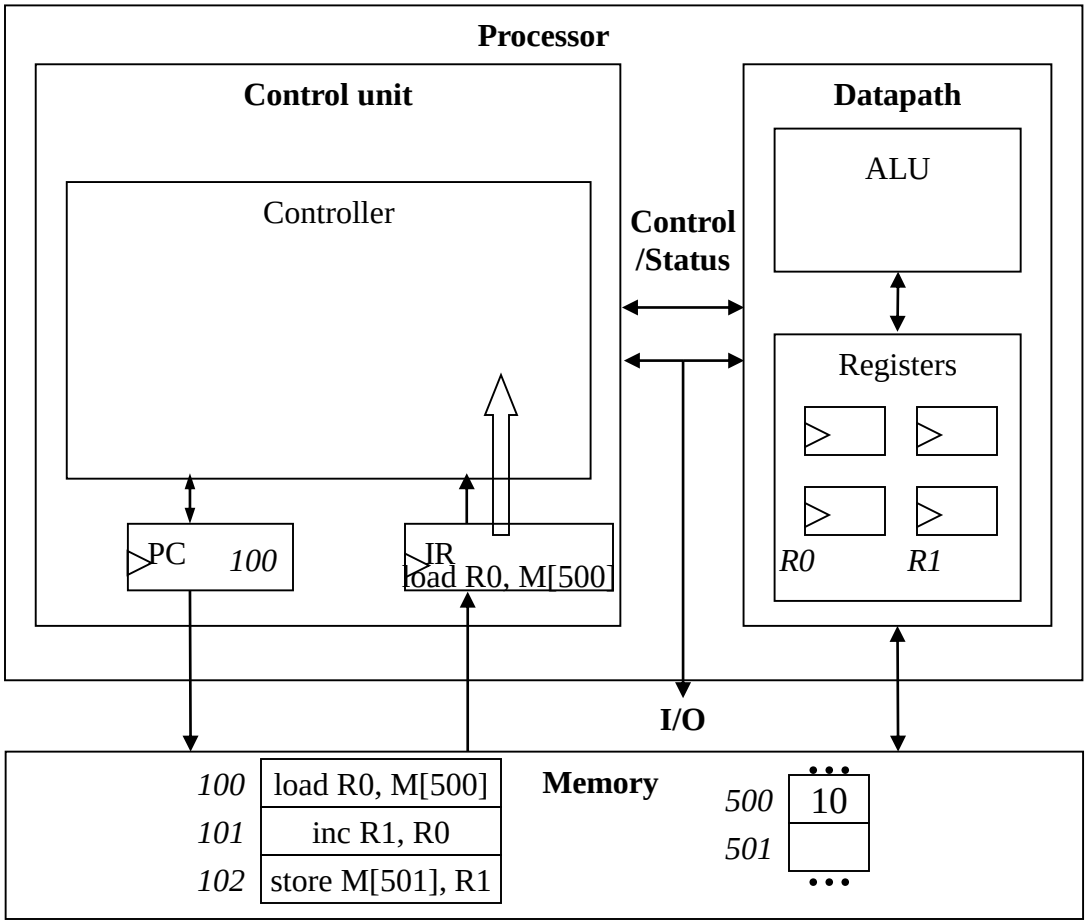
Control Unit Sub-Operations

- Fetch
  - Get next instruction into IR
  - PC: program counter, always points to next instruction
  - IR: holds the fetched instruction



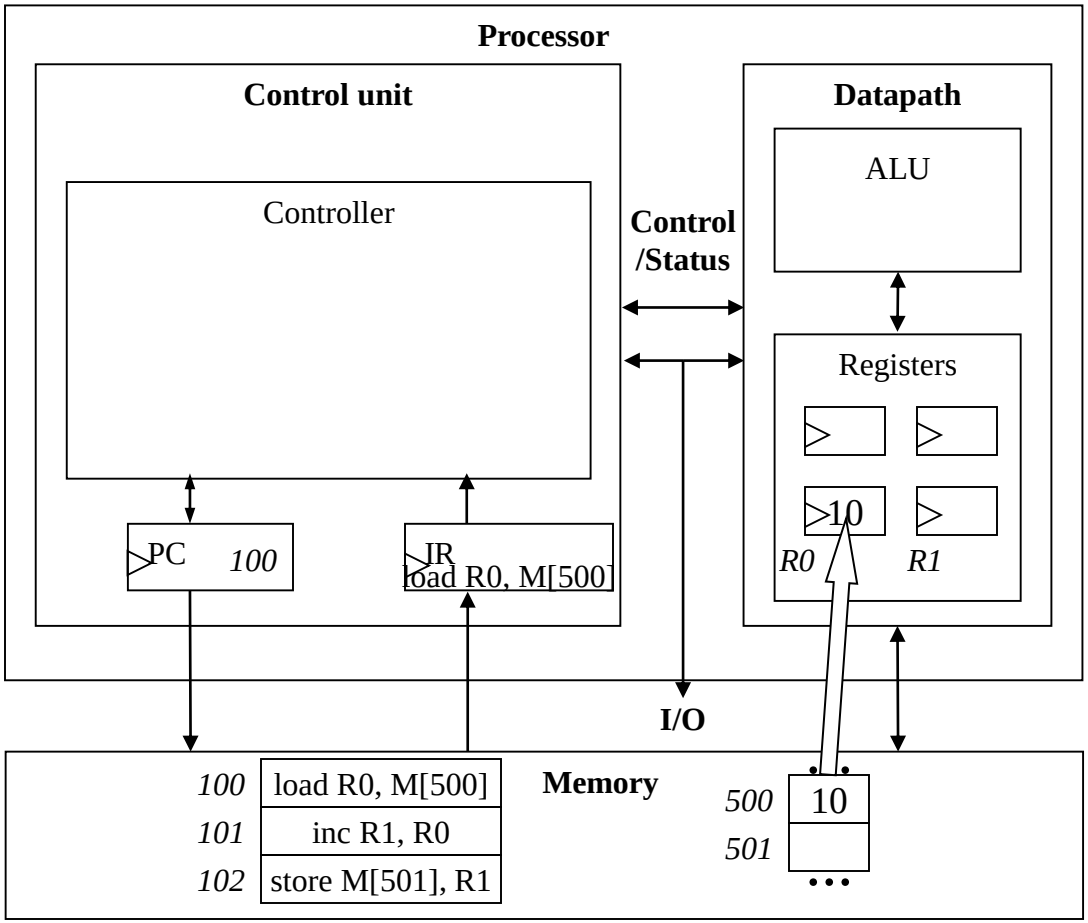
Control Unit Sub-Operations

- Decode
  - Determine what the instruction means



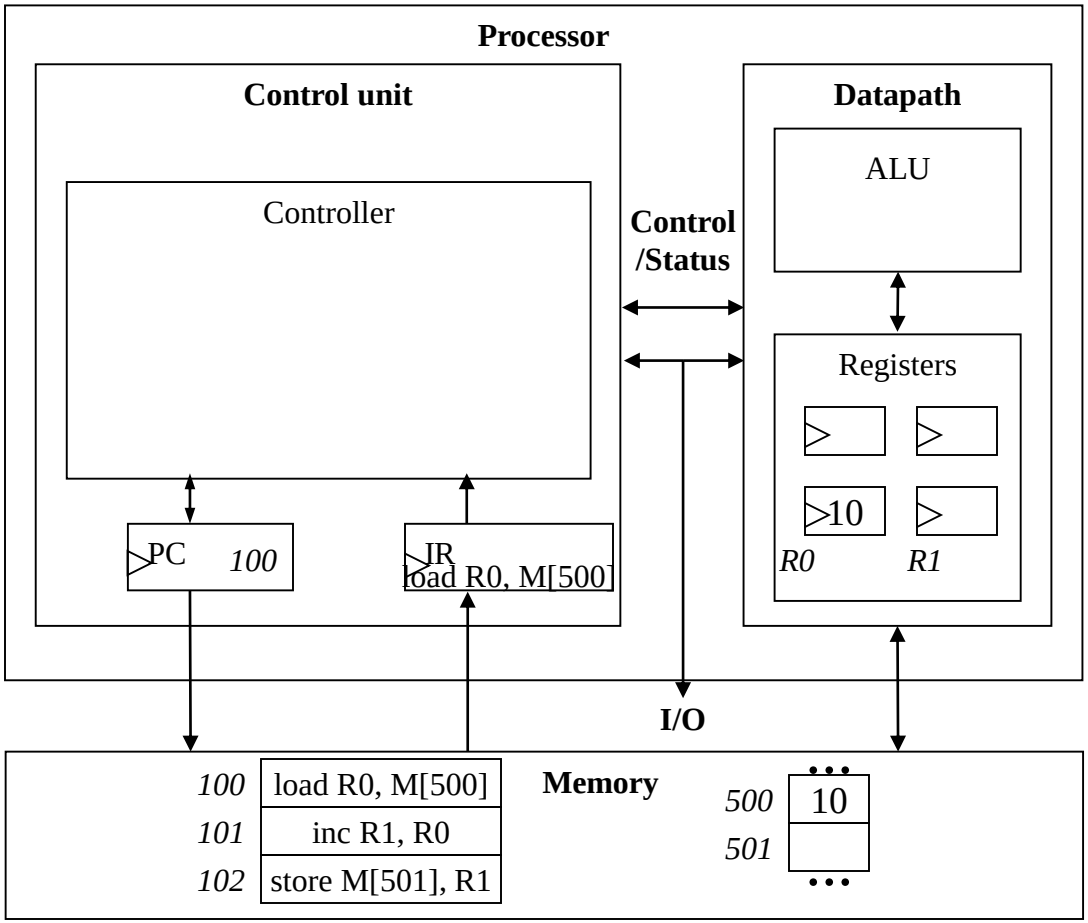
Control Unit Sub-Operations

- Fetch operands
  - Move data from memory to datapath register



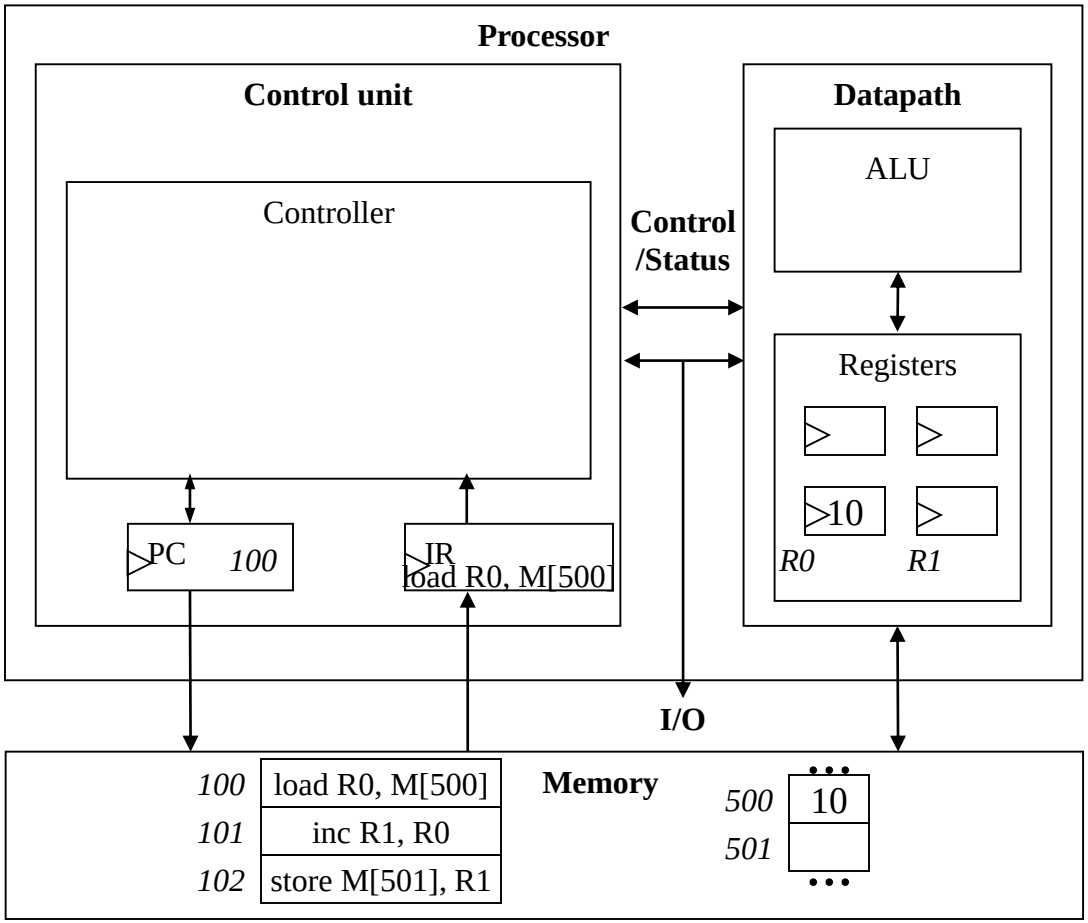
Control Unit Sub-Operations

- Execute
  - Move data through the ALU
  - (This example instruction does nothing during this sub-operation)

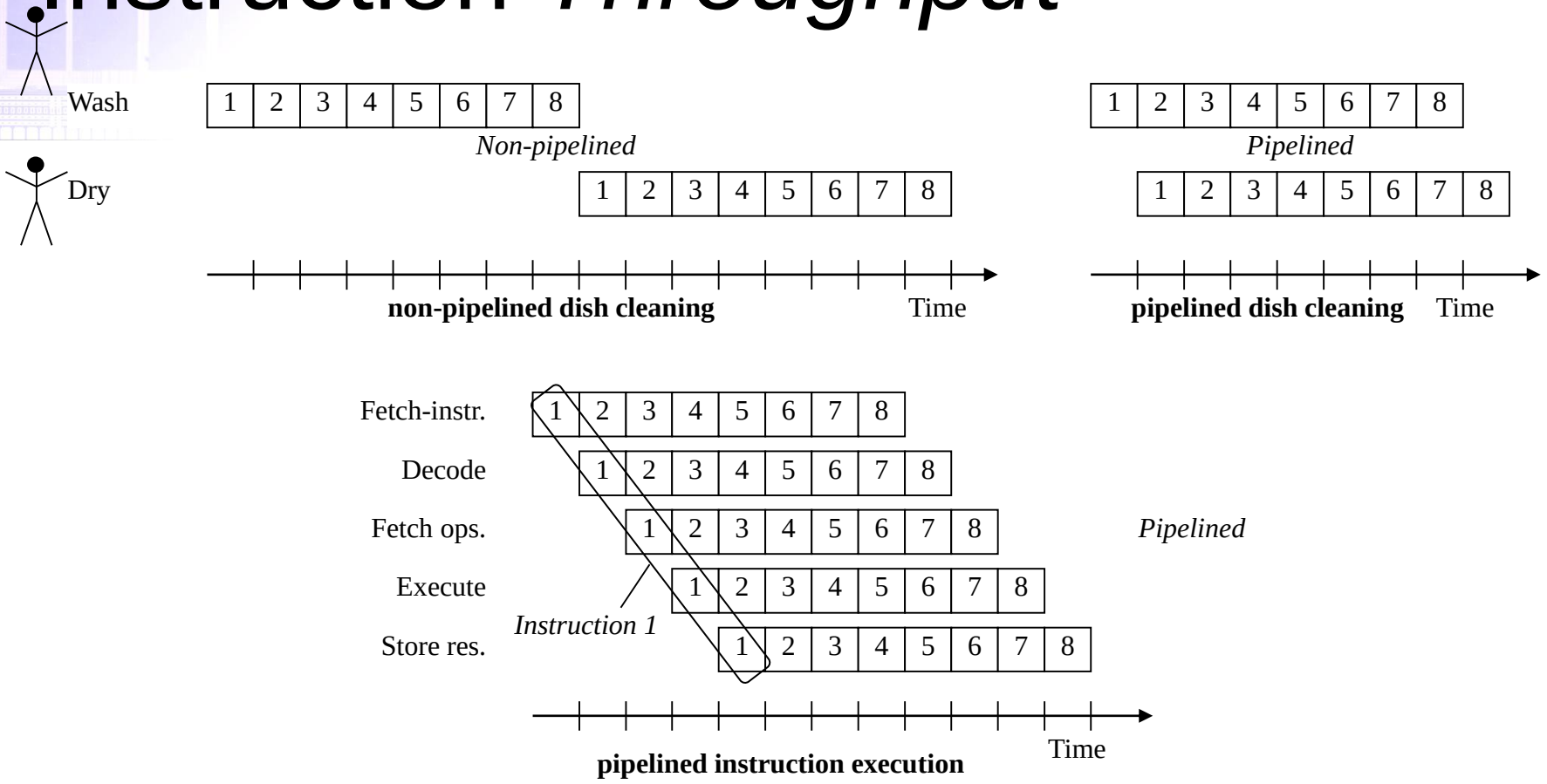


Control Unit Sub-Operations

- Store results
  - Write data from register to memory
  - (This example instruction does nothing during this sub-operation)



Pipelining: Increasing Instruction *Throughput*



Superscalar and VLIW Architectures

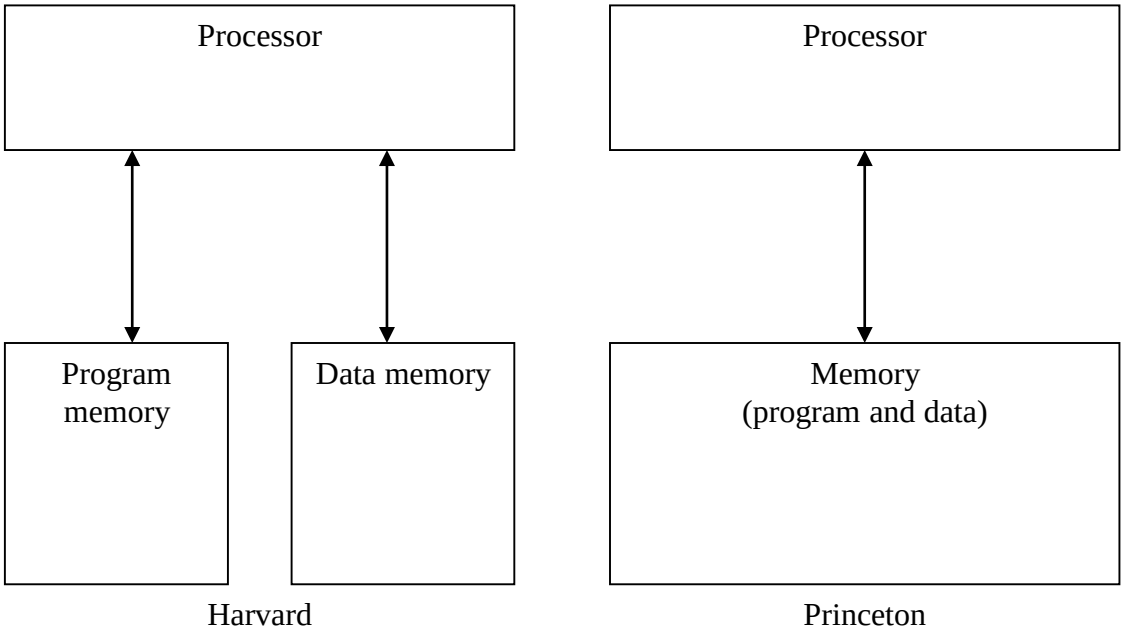
- Performance can be improved by:
  - Faster clock (but there's a limit)
  - Pipelining: slice up instruction into many stages
  - *Multiple ALUs* to support more than one instruction stream – superscalar

Superscalar and VLIW Architectures

- Superscalar
  - Scalar: non-vector operations
  - Fetches instructions in batches, executes as many as possible
    - May require extensive hardware to detect independent instructions (dynamic)
    - VLIW (Very-Long Instruction Word): each word in memory has multiple independent instructions (static)
      - Relies on the compiler to detect and schedule instructions
      - Currently growing in popularity

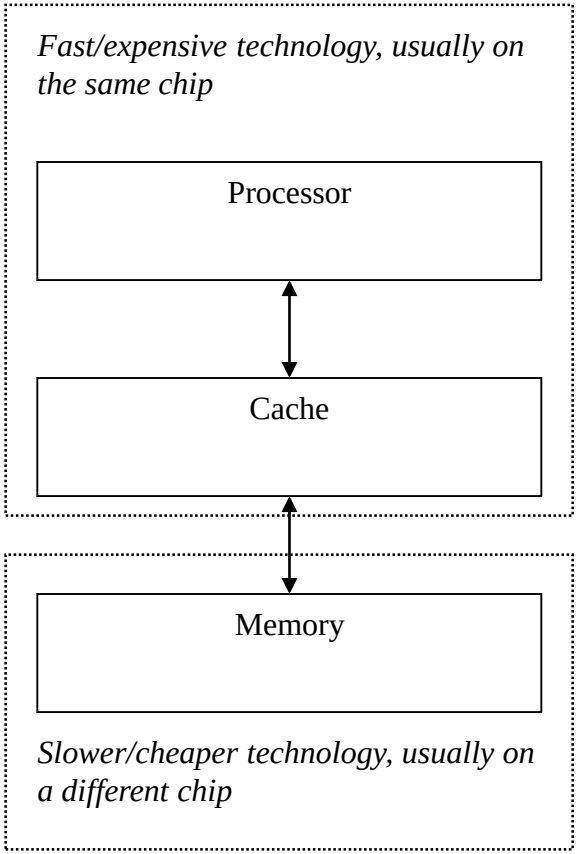
Two Memory Architectures

- Princeton
  - Fewer memory wires
- Harvard
  - Simultaneous program and data memory access



Cache Memory

- Memory access may be slow
- Cache is small but fast memory close to processor
  - Holds copy of part of memory
  - Hits and misses



Programmer's View

- Programmer doesn't need understanding of architecture in detail
  - Instead, needs to know what instructions can be executed
- Two levels of instructions:
  - Assembly level
  - Structured languages (C, C++, Java, etc.)
- Most developments today are done using structured languages
  - But, some assembly level programming may still be necessary
  - Drivers: portion of program that communicates with and/or controls (drives) another device

Assembly-Level Instructions

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

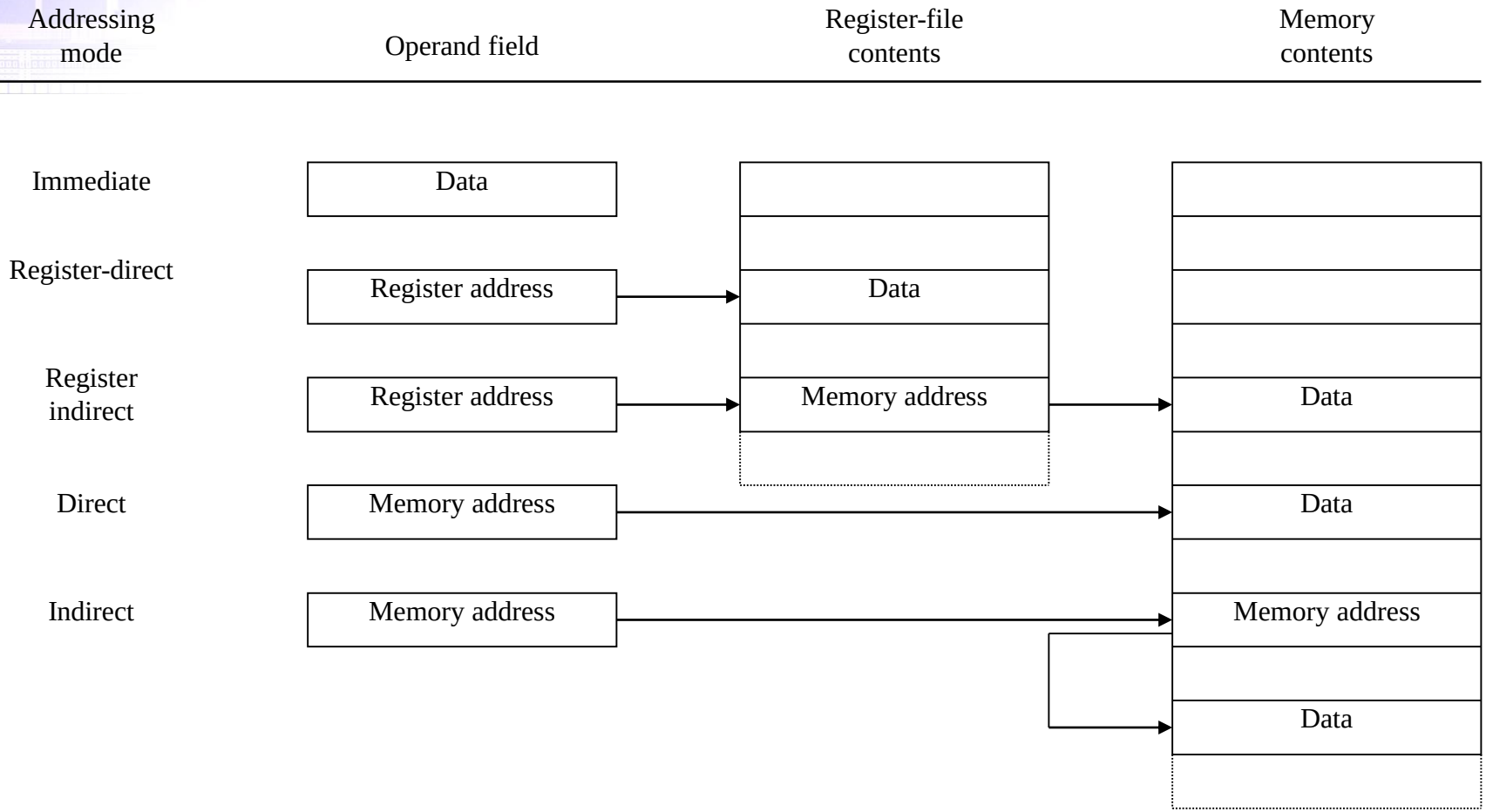
- Instruction Set
  - Defines the legal set of instructions for that processor
    - Data transfer: memory/register, register/register, I/O, etc.
    - Arithmetic/logical: move register through ALU and back
    - Branches: determine next PC value when not just PC+1

A Simple (Trivial) 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)

opcode operands

Addressing Modes



Programmer Considerations

- Program and data memory space
  - Embedded processors have often very limited memory
    - e.g., 64 Kbytes program, 256 bytes of RAM (expandable)
- Registers: How many are there?
  - Only a direct concern for assembly-level programmers
  - Be aware of special-function registers

Programmer Considerations

- I/O
  - Two ways
  - I/O instructions for parallel I/O of CPU
    - Such as Intel x86
  - Memory-mapped I/O (through system bus)
    - The most common way

Programmer Considerations

- Interrupts
  - Cause the processor to suspend execution of the main program and jump to an **interrupt service routine (ISR)** or **interrupt handler**
  - After the ISR completes, the processor resumes execution of the **main program (foreground program)** by restoring the PC
  - The ISR should be located at a specific address in program memory

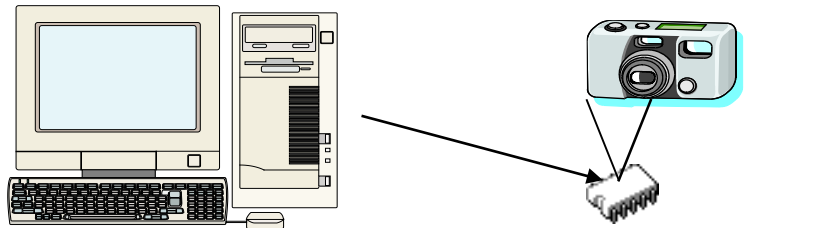
Operating System

- Optional software layer providing low-level services to a program (application).
  - Resource (CPU, memory, …) management
  - File management, disk access
  - Keyboard/display interfacing
  - Scheduling multiple programs for execution
    - Or even just multiple threads from one program
  - Program makes system calls to the OS

```
DB file_name "out.txt" -- store file name

MOV R0, 1324           -- system call "open" id
MOV R1, file_name      -- address of file-name
INT 34                 -- cause a system call
JZ  R0, L1             -- if zero -> error

   . . . read the file
JMP L2                 -- bypass error cond.
L1:
   . . . handle the error

L2:
```

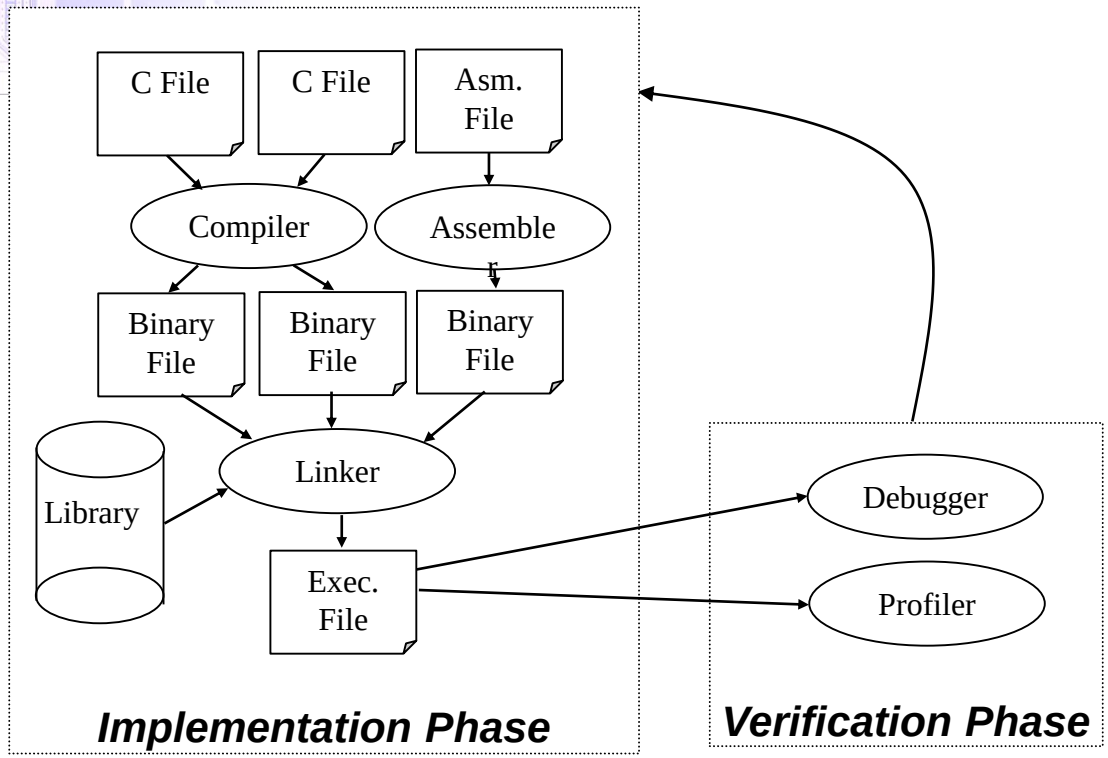
Development Environment

- Development processor (host)
  - The processor on which we write and debug our programs
    - Usually a PC
- *Target processor*
  - The processor that the program will run on in our embedded system
    - Often different from the development processor



Development processor Target processor

Software Development Process



- Typically, these tools are combined into a single integrated development environment (IDE)
- Compilers
  - **Cross compiler**
    - Runs on one processor, but generates code for another
- Assemblers
- Linkers
- Debuggers
- Profilers

Software Development Process

- Debugger
  - Run on the development processor
  - Support stepwise program execution
  - Support breakpoints
  - When the program stops, the user can examine values of various memory and register location
  - Source-level debuggers: step-by-step in source program
  - **Use instruction-set simulators (ISS)** or virtual machines (VM)

Software Development Process

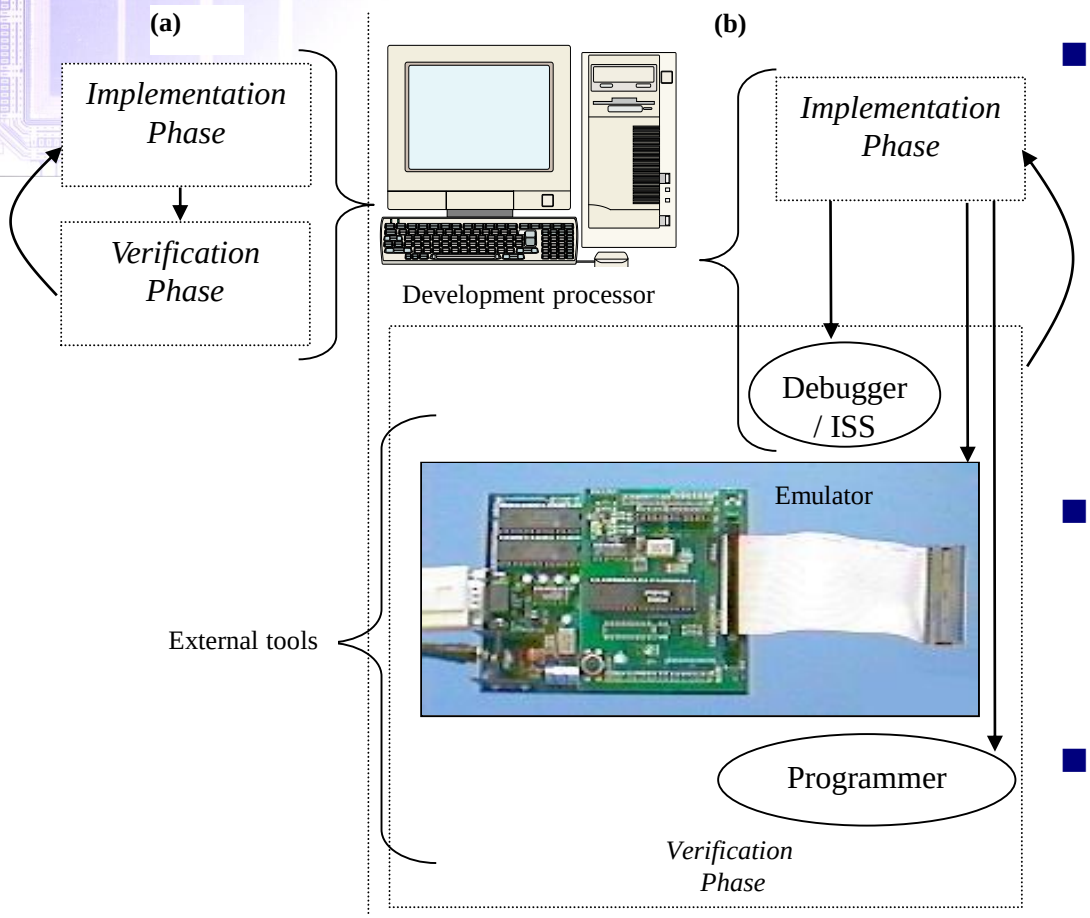
- Emulator
  - Support debugging of the program while it executes on the target processor
  - Microprocessor in-circuit emulator (ICE)
    - A special hardware tool to emulate the behavior of a processor

Instruction Set Simulator For a Simple Processor

```
#include <stdio.h>
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;
         case 4: reg[fb & 0x0f] += reg[sb >> 4]; break;
         case 5: reg[fb & 0x0f] -= reg[sb >> 4]; break;
         case 6: pc += sb; break;
         default: return -1;
```

```
}
   }
   return 0;
}

int main(int argc, char *argv[]) {

   FILE* ifs;

   If( argc != 2 ||
       (ifs = fopen(argv[1], "rb") == NULL ) {
            return -1;
   }
   if (run_program(fread(program,
       sizeof(program) == 0) {
            print_memory_contents();
            return(0);
   }
   else return(-1);
}
```

Testing and Debugging



- ISS
  - Gives us control over time – set breakpoints, look at register values, set values, step-by-step execution, ...
  - But, doesn't interact with real environment
- Download to board
  - Use device programmer
  - Runs in real environment, but not controllable
- Compromise: emulator
  - Runs in real environment, at real-time speed or near
  - Supports some controllability from the PC

Software Development Process

- Conventional stages of development
  - Debugging using an ISS
  - Emulation using an emulator
  - Field testing by downloading the program directly into the target processor
- Different levels of simulation
  - Instruction-level simulator
  - Cycle-level simulator
  - Hardware/software co-simulator

CPU Power Consumption

- Most modern CPUs are designed with power consumption in mind to some degree

- Voltage drops: power consumption proportional to V^2
- Toggling: more activity means more power
- Leakage: basic circuit characteristics; can be eliminated by disconnecting power

CPU Power-Saving Strategies

- Reduce power supply voltage
- Run at lower clock frequency
- Disable function units with control signals when not in use
- Disconnect parts from power supply when not in use

Power management styles

- Static power management: does not depend on CPU activity.
  - Example: user-activated power-down mode.
- Dynamic power management: based on CPU activity.
  - Example: disabling off function units.
  - Dynamic Voltage Frequency Scaling (DVFS)

Application: PowerPC 603 Energy Features

- Provides doze, nap, sleep modes.
- Dynamic power management features:
  - Uses static logic.
  - Can shut down unused execution units.
  - Cache organized into subarrays to minimize amount of active circuitry.

PowerPC 603 Activity

- Percentage of time when units are idle for SPEC integer/floating-point:

unit	Specint92	Specfp92
D cache	29%	28%
I cache	29%	17%
load/store	35%	17%
fixed-point	38%	76%
floating-point	99%	30%
system register	89%	97%

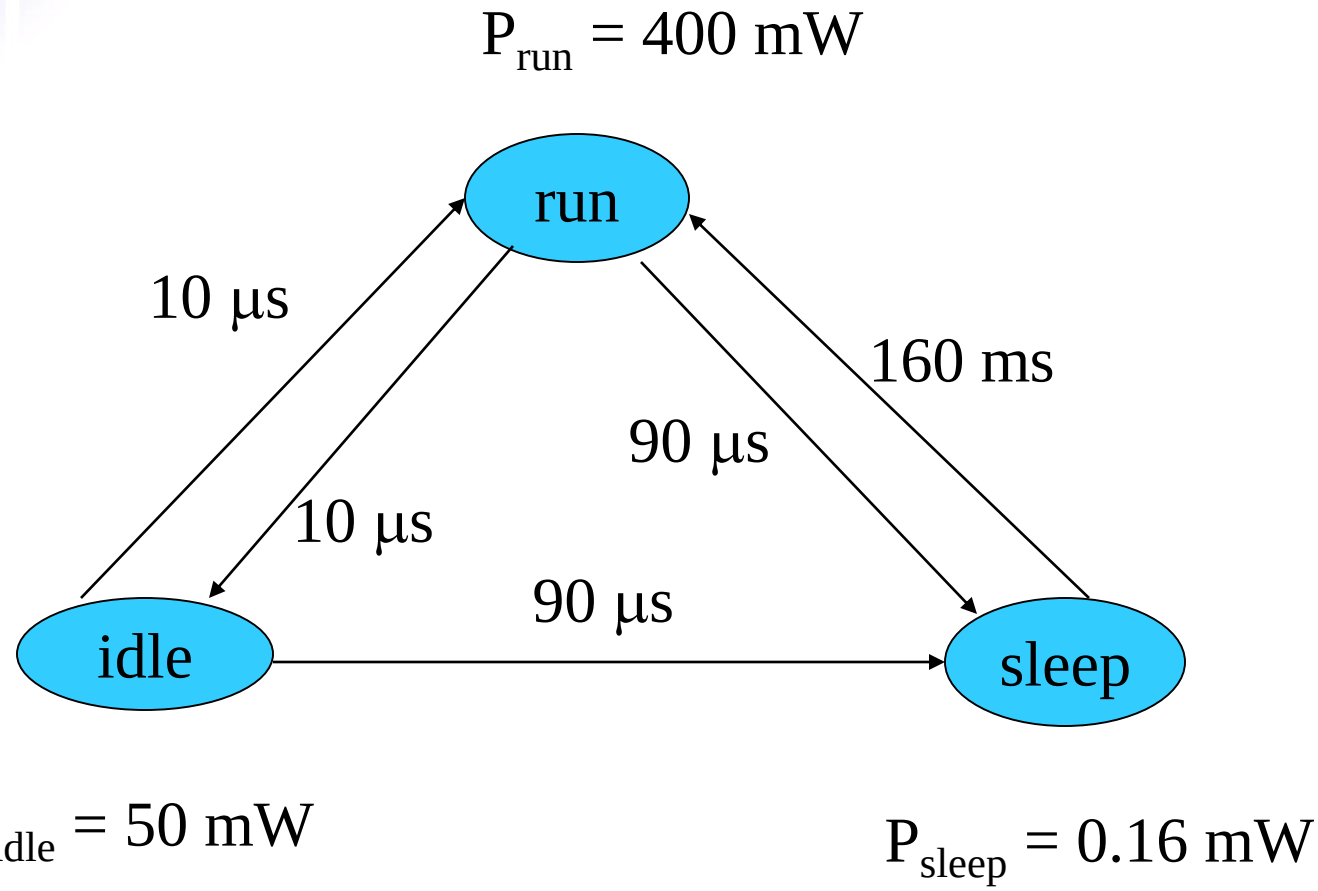
Power-Down Costs

- Going into a power-down mode costs:
  - time;
  - energy.
- Must determine if going into mode is worthwhile.
- Can model CPU power states with power state machine.

Application: StrongARM SA-1100 Power Saving

- Processor takes two supplies:
  - VDD is main 3.3V supply.
  - VDDX is 1.5V.
- Three power modes:
  - Run: normal operation.
  - Idle: stops CPU clock, with logic still powered.
  - Sleep: shuts off most of chip activity; 3 steps, each about 30 μs; wakeup takes > 10 ms.

SA-1100 Power State Machine



Selecting a Microprocessor

- Issues
  - Technical: speed, power, size, cost
  - Other: development environment, prior expertise, licensing, etc.
- Speed: how to evaluate a processor's speed?
  - Clock speed – but instructions per cycle may differ
  - Instructions per second – but work per instr. may differ
  - **Dhrystone**: Synthetic benchmark, developed in 1984. Dhrystones/sec.
    - MIPS: 1 MIPS = 1757 Dhrystones per second (based on Digital's VAX 11/780). A.k.a. Dhrystone MIPS. Commonly used today.
      - So, 750 MIPS = 750*1757 = 1,317,750 Dhrystones per second
  - SPEC: set of more realistic benchmarks, but oriented to desktops
  - EEMBC – EDN Embedded Benchmark Consortium, www.eembc.org
    - Suites of benchmarks: automotive, consumer electronics, networking, office automation, telecommunications

General Purpose Processors

Processor	Clock speed	Periph.	Bus Width	MIPS	Power	Trans.	Price
General Purpose Processors							
Intel PIII	1GHz	2x16 K L1, 256K L2, MMX	32	~900	97W	~7M	$900
IBM PowerPC 750X	550 MHz	2x32 K L1, 256K L2	32/64	~1300	5W	~7M	$900
MIPS R5000	250 MHz	2x32 K 2 way set assoc.	32/64	NA	NA	3.6M	NA
StrongARM SA-110	233 MHz	None	32	268	1W	2.1M	NA
Microcontroller							
Intel 8051	12 MHz	4K ROM, 128 RAM, 32 I/O, Timer, UART	8	~1	~0.2W	~10K	$7
Motorola 68HC811	3 MHz	4K ROM, 192 RAM, 32 I/O, Timer, WDT, SPI	8	~.5	~0.1W	~10K	$5
Digital Signal Processors							
TI C5416	160 MHz	128K, SRAM, 3 T1 Ports, DMA, 13 ADC, 9 DAC	16/32	~600	NA	NA	$34
Lucent DSP32C	80 MHz	16K Inst., 2K Data, Serial Ports, DMA	32	40	NA	NA	$75

Sources: Intel, Motorola, MIPS, ARM, TI, and IBM Website/Datasheet; Embedded Systems Programming, Nov. 1998

Application-Specific Instruction-Set Processors (ASIPs)

- General-purpose processors
  - Sometimes too general to be effective in demanding applications
    - e.g., video processing – requires huge video buffers and operations on large arrays of data, inefficient on a GPP
  - But single-purpose processor has high NRE, not programmable
- ASIPs – targeted to a particular domain
  - Contain architectural features specific to that domain
    - e.g., embedded control, digital signal processing, video processing, network processing, telecommunications, etc.
  - Still programmable

A Common ASIP: Microcontroller

- For embedded control applications
  - Reading sensors, setting actuators
  - Mostly dealing with events (bits): data is present, but not in huge amounts
  - e.g., VCR, disk drive, digital camera, washing machine, microwave oven
- Microcontroller features
  - On-chip peripherals
    - Timers, analog-digital converters, serial communication, etc.
    - Tightly integrated for programmer, typically part of register space
  - On-chip program and data memory
  - Direct programmer access to many of the chip's pins
  - Specialized instructions for bit-manipulation and other low-level operations

Co-Processor

- Co-processor: added function unit that is called by instruction
  - Floating-point units are often structured as co-processors
- Tightly-coupled to the CPU
- When receiving a co-processor instruction, the CPU must activate the co-processor and pass it the relevant instructions
- Co-processor: can load/store co-processor registers and CPU registers
- To provide compatibility, the function of co-processor can be emulated with software interrupt handler
- ARM allows up to 16 designer-selected co-processors.

Another Common ASIP: Digital Signal Processors (DSP)

- For signal processing applications
  - Large amounts of digitized data, often streaming
  - Data transformations must be applied fast
  - e.g., cell-phone voice filter, digital TV, music synthesizer
- DSP features
  - Several instruction execution units
  - Multiply-accumulate single-cycle instruction, other instrs.
  - Efficient vector operations
    - e.g., add two arrays vector ALUs, loop buffers, etc.

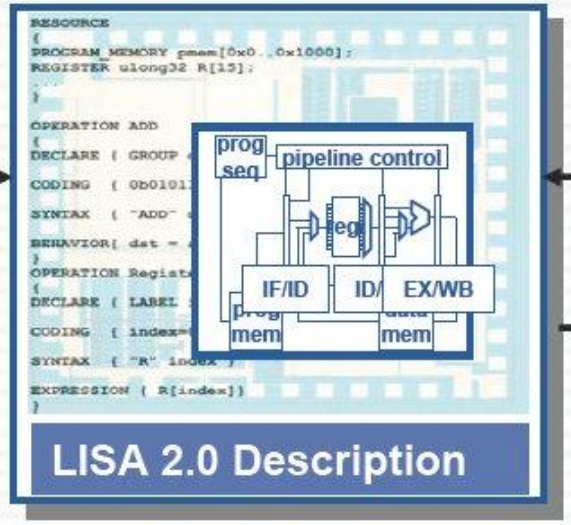
Trend: Even More Customized ASIPs

- In the past, microprocessors were acquired as chips
- Today, we increasingly acquire a processor as Intellectual Property (IP)
  - e.g., synthesizable VHDL model
- Opportunity to add a custom datapath hardware and a few custom instructions, or delete a few instructions
  - Can have significant performance, power and size impacts

Trend: Even More Customized ASIPs

- Problem: need compiler/debugger for customized ASIP
  - Remember, most development uses structured languages
  - One solution: automatic compiler/debugger generation
    - e.g., www.tensillica.com (acquired by Cadence)
  - Another solution: retargettable compilers
  - Modern solution: automatic hardware/compiler/debugger generation with a processor architecture design language
    - CoWare LISATek ➔ Synopsys Processor Designer
    - ARM MaxCore

Architecture Exploration – An iterative Design Flow



Software

C-Compiler

Assembler

Linker

Multi-dimensional design space!

Model Verification & Evaluation

Evaluation Results
Profile Information, Application Performance

Integrate into SoC Environment

RTL Model
+
Manual refinement

Model Verification & Evaluation

Evaluation Results
Chip Area, Clock Speed, Power Consumption

Hardware

CoWare™

DVB-T and Application Specific Multirate DSP

Infineon Low-Power, DVB-T Single-Chip Receiver:

- ASIP for DVB-T acquisition and tracking algorithms
- Harvard Architecture
- 60 mostly RISC-like Instructions
- 8x32-Bit General Purpose Registers, 4x9-Bit Address Registers
- 2048x20-Bit Instruction ROM, 512x32-Bit Data Memory

Infineon Application Specific Multirate DSP

- ASIP for interpolation and decimation filters or CORDIC
- Highly optimized data kernel
- Complex FSM to control
- Success story at http://www.coware.com

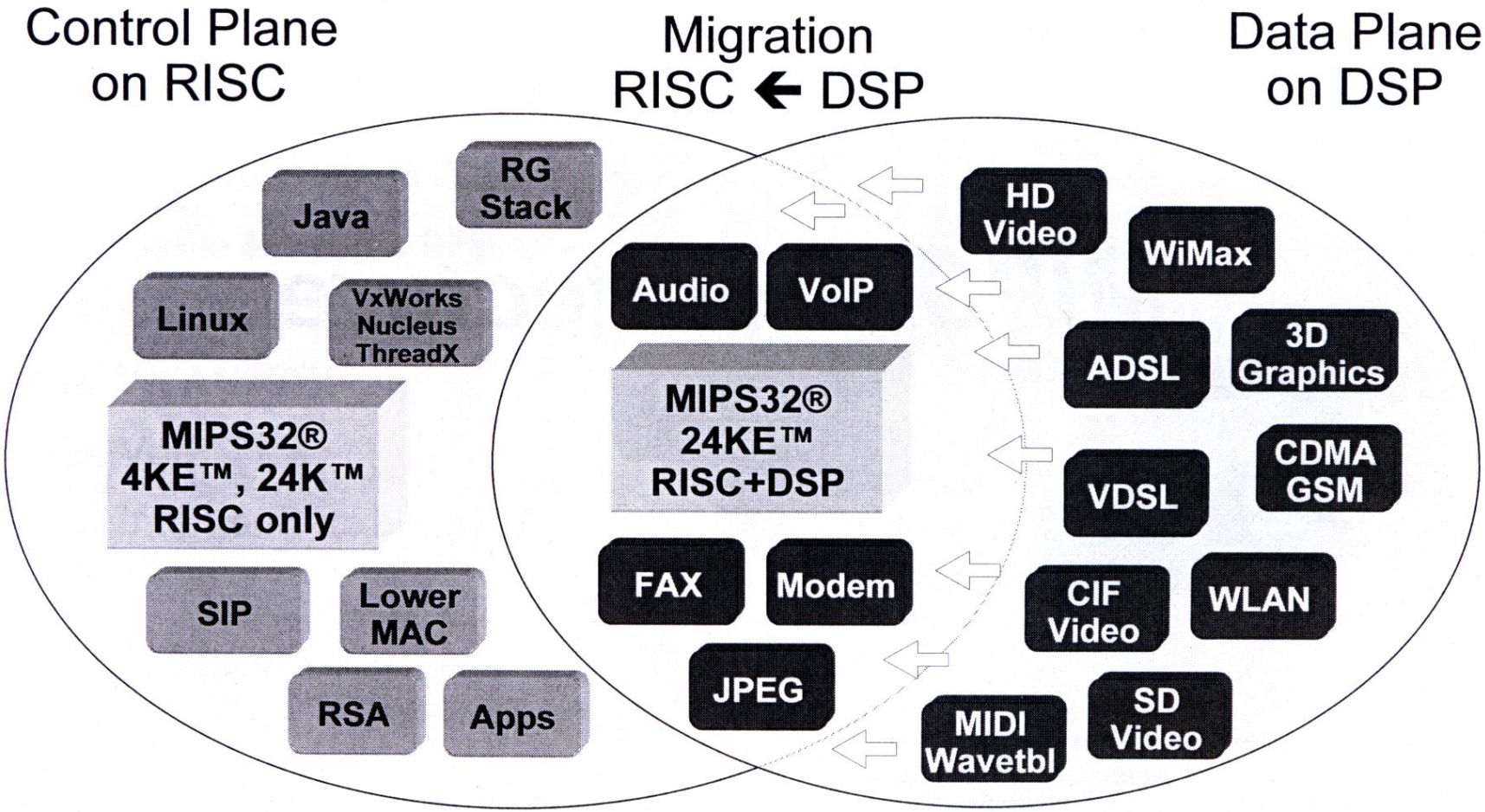
Comparisons with reference models

	Speed	Area
Infineon ASMD	5.09 ns	11 678 Gates
…handwritten and optimized VHDL	4.22 ns	9 549 Gates
Infineon ICORE	8 ns	59 000 Gates
…handwritten and optimized VHDL	8 ns	58 473 Gates

➔ **Design Time reduction: up to 60 %**

CoWare™

Signal Processing is Migrating to the Host

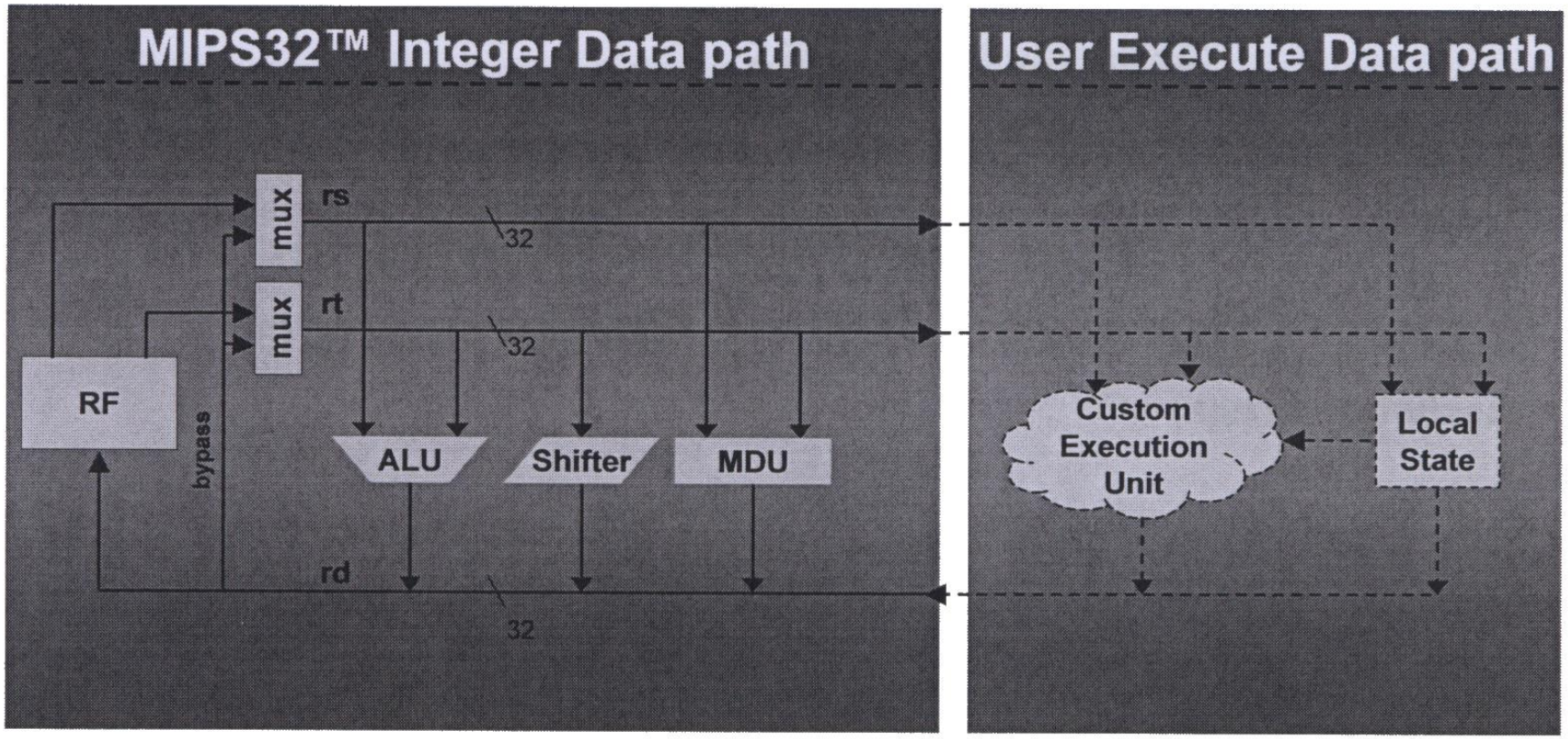


CoreExtend

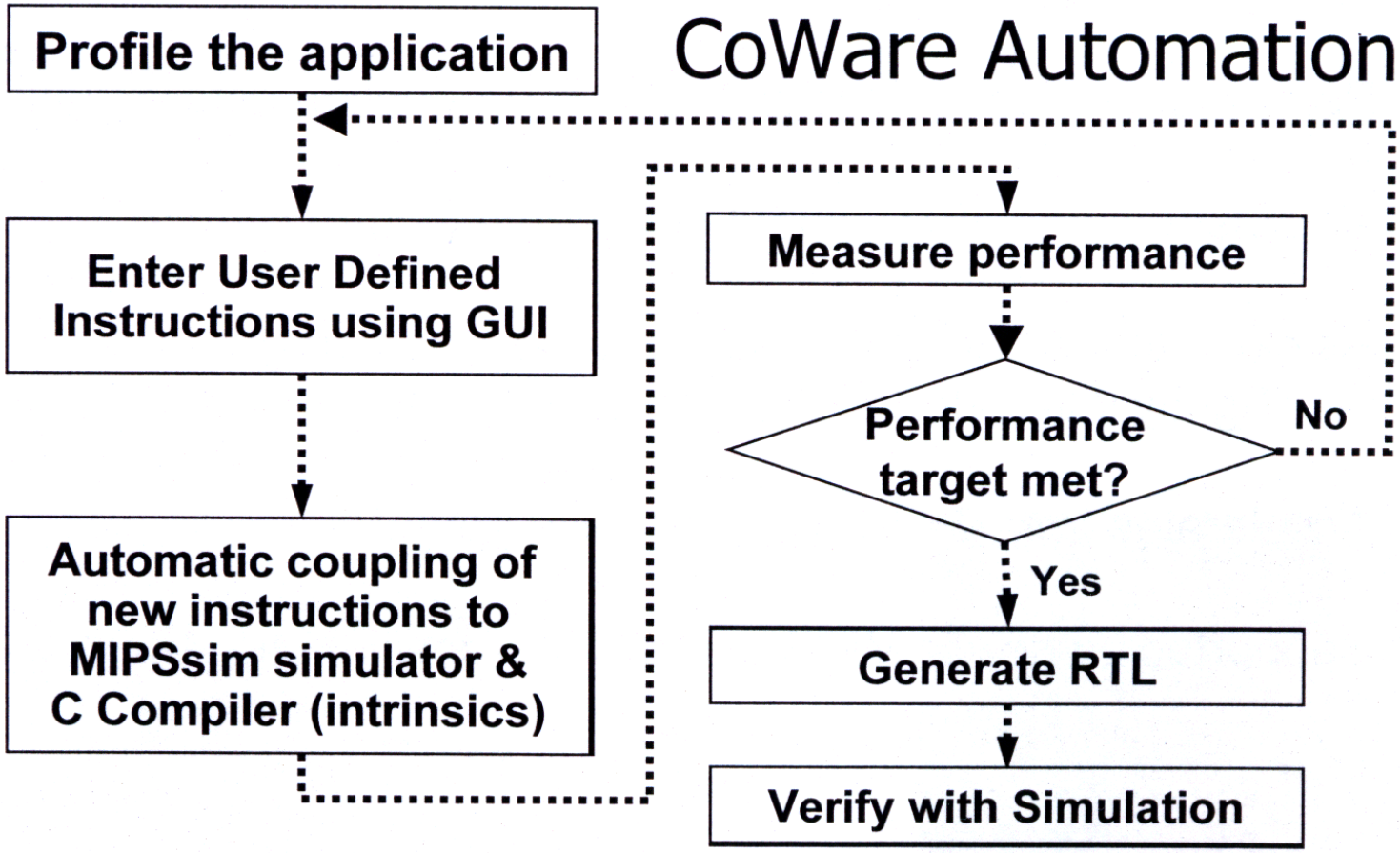
– User Defined Instructions (UDI) for even Higher Performance

- Performance tuning, design reuse and production differentiation
- Add instruction (UDI) without architecture license
- As close to the microprocessor core as you can get
  - Tightly integrated to pipeline and the GPR
- Accelerates from 2x to hardwired-like speed
- Supported by newer MIPS cores: 4KE family, M4K, 4KSd, 24K

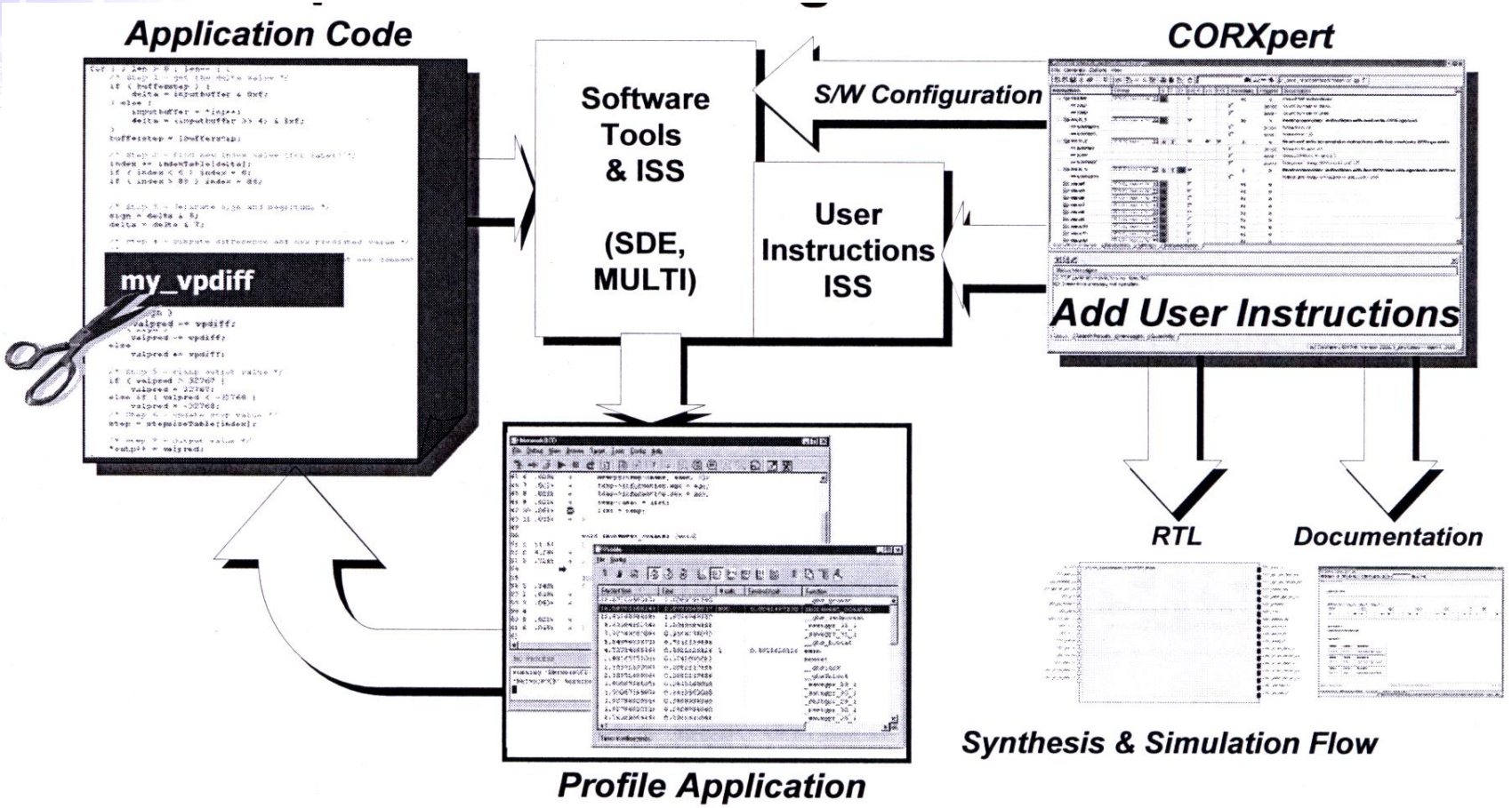
Users Execute Block Diagram



CorExtend™ Instruction Development Flow



CORXpert – Automating CorExtend™ UDIs

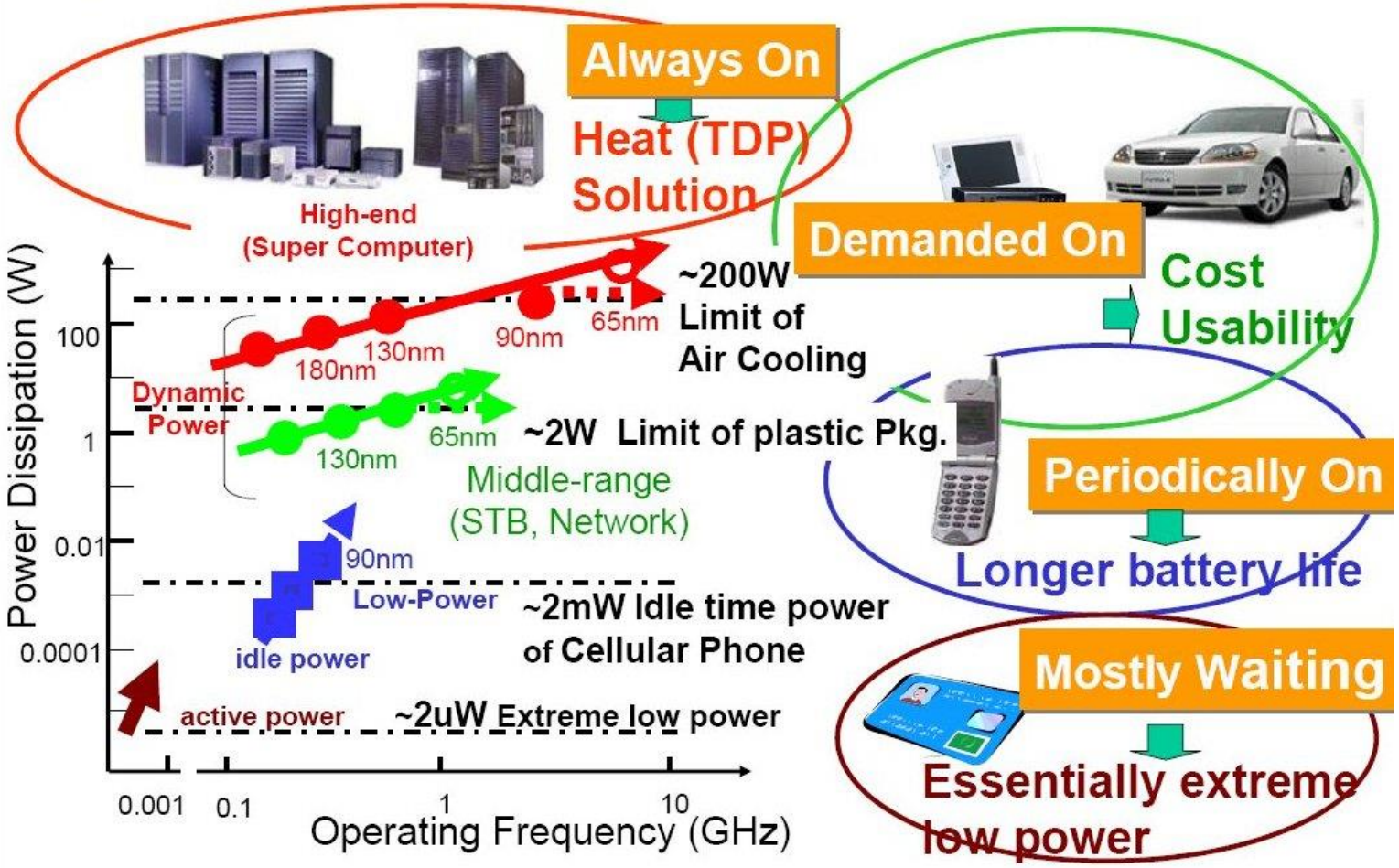


CorExtend/UDI Application Examples

- VoIP simple example
  - 2X total speed up over optimized code with 7K gates
- 802.11a/b/g/l/e lower MAC high throughput wire-speed examples
  - 30X AES (128-bit key) speed up with 10.5K gates + 20x64bit round key RAM
- ADSL2 + SIMD reuse example
  - >=40X RS decode speed up over optimized code with 8K gates
- JPEG decode acceleration

Trend: Multi-core
Comes from Low Power Demands

- Low power : key technology in any application domain



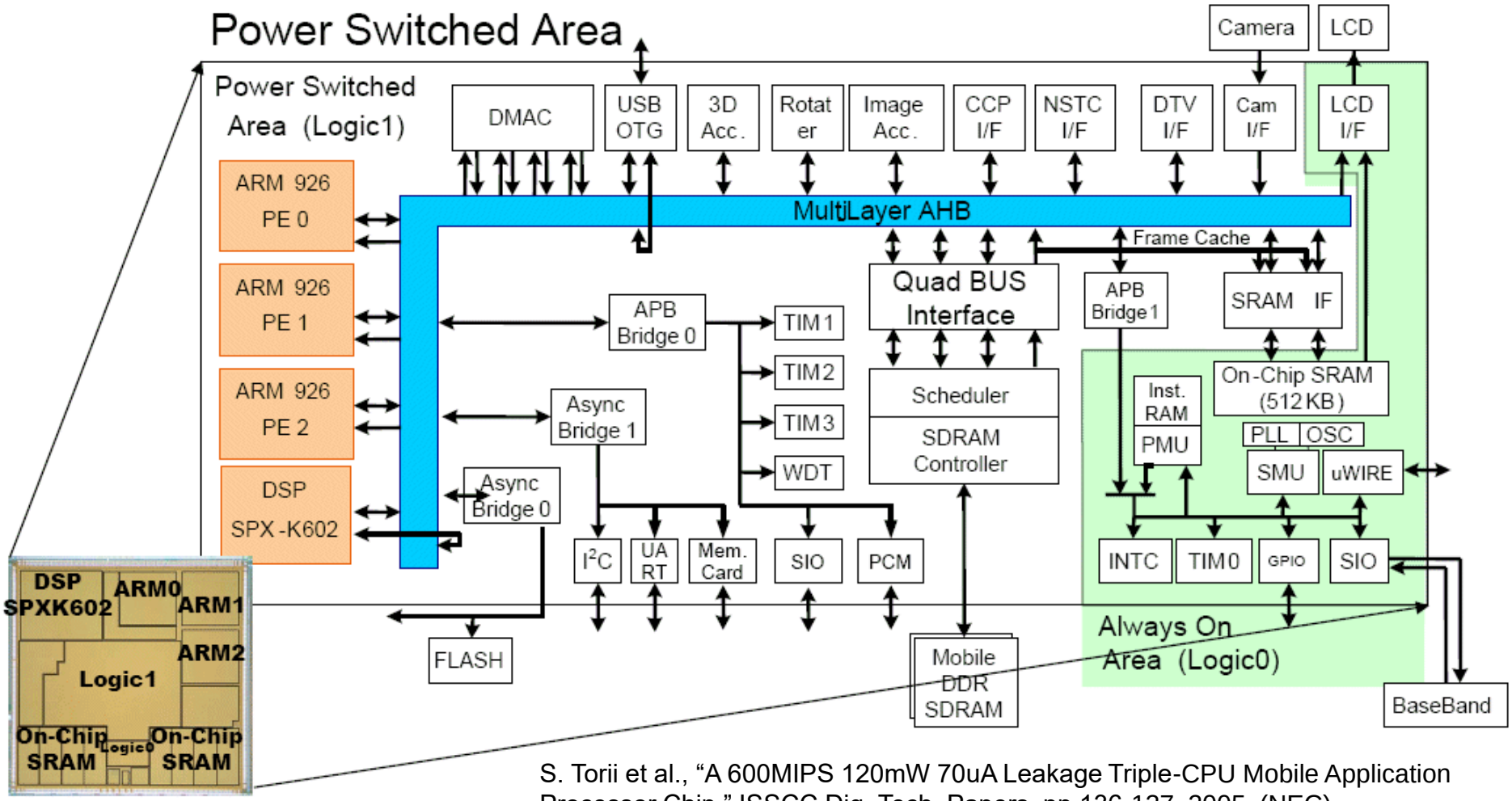
Satoshi Matsushitam, "Low Power Multi-Core Chips for Mobile Embedded Applications," Mlti-core Processor Forum Notes, ISSCC2006.

Solution Space in each Category

Category	Apps	Characteristics	Throughput technique	Low power technique	Products
Always On	High-end server	Independent transactions	Clock, ILP, Parallel processing	-	IBM Power4
	Server or Numerical	Independent transactions or calculations	Resource sharing by multiple threads	Low Vdd	Sun Niagara, Tera MTA
	Network	Independent packets	Parallel processing	-	Broadcom BCM1480, Cavium Octeon, PMC-Sierra RM11200
Demanded On	Media	Independent 3D or streaming data	Clock, SIMD, Parallel processing	-	IBM/Sony/Toshiba Cell
	Desktop	General-purpose, Media	Clock, ILP, SIMD, Parallel processing	Low Vdd	Intel Pentium D, AMD Athlon 64 x2
Periodic-ally On	Mobile	General-purpose, Media	Parallel processing, SIMD, DSP, HW engine	Low Vdd, DVFS, Power-off	NEC MP211, NEC MPCore, TI OMAP, Hitachi SH-Mobile
Mostly Waiting	Security	Encrypt, Decrypt	HW engine	Device, Small logic, Slow clock	Infineon 66Plus, Sony Felica

Mobile Application SoC: MP211

- Asymmetric Multi-processor (AMP) which integrates three ARM926 (200MHz) and a DSP
- System level power control for intermittent load



S. Torii et al., "A 600MIPS 120mW 70uA Leakage Triple-CPU Mobile Application Processor Chip," ISSCC Dig. Tech. Papers, pp.136-137, 2005. (NEC)

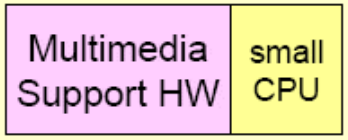
Progress of Embedded Platforms

Solution to SW/HW Productivity & Performance

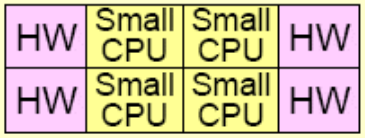
→ Scalable solution
with regular multi-core structure

Now
Consequent multi-core

Type A

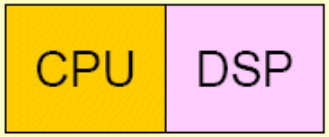


Type B



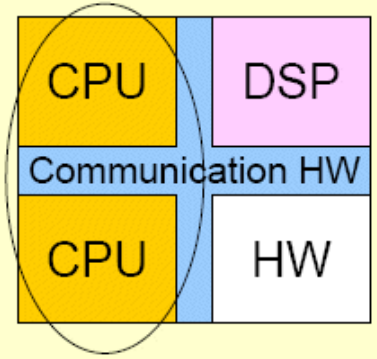
Type C

Type D

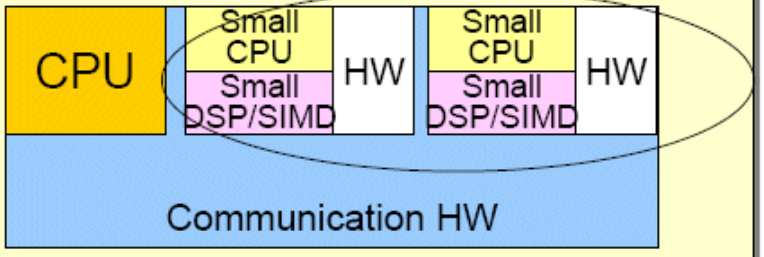


Next generation

Type E: CPU parallel



Type F: Media parallel



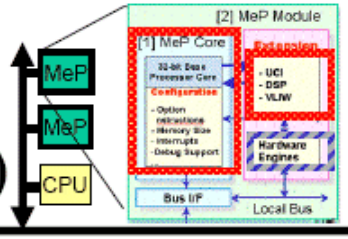
Scalability Enhancement with Multi-core

Small
Flexibility
High

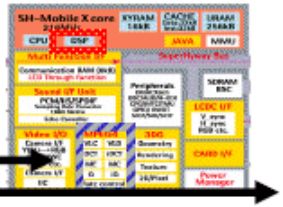
HW Accelerator
DSP
Media Parallel
SIMD
CPU Parallel

UniPhier (Matsushita)

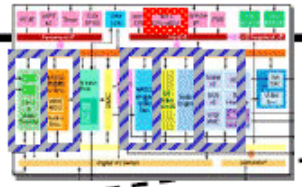
MeP (Toshiba)



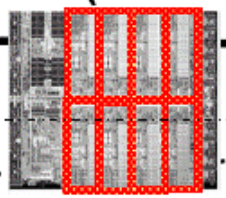
SH-Mobile (Renesas)



EMMA (NECEL)

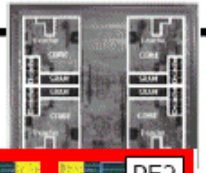


Cell (IBM/Sony/Toshiba)

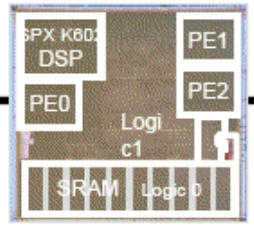


OMAP (TI)

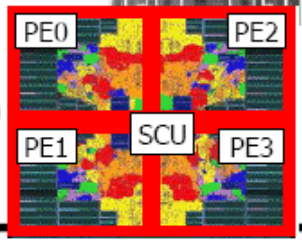
FR-V (Fujitsu)



MP211 (NEC)



MPCore (ARM/NEC/Nvidia)



High Performance — Performance Range — Low Power

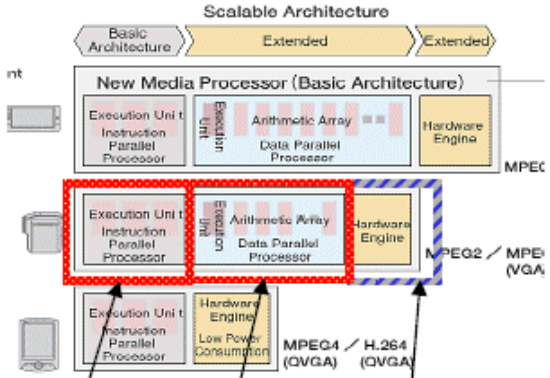





Media Parallel Multi-cores

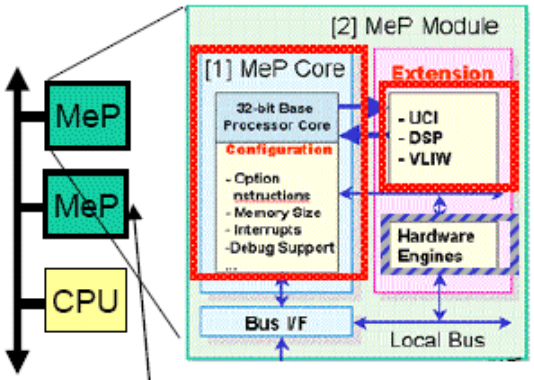
- Various approaches for scalability

UniPhier (Matsushita)



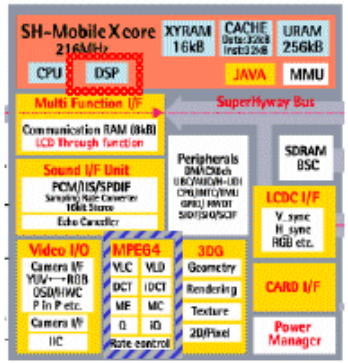
(VLIW + SIMD + HW) + CPU

MeP (Toshiba)



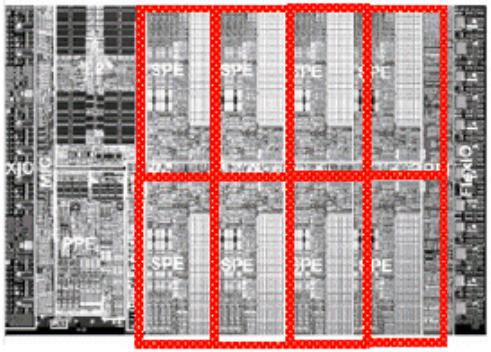
MediaEngine x N + CPU

SH-Mobile (Renesas)



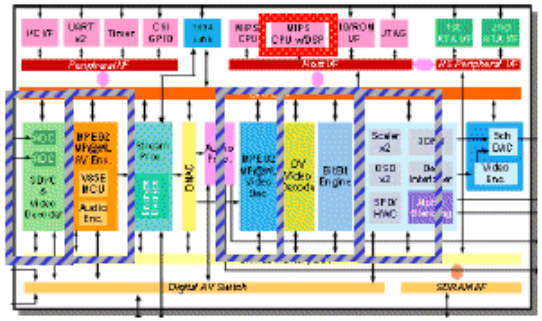
DSP+ HW Engine + CPU

Cell (IBM/Sony/Toshiba)



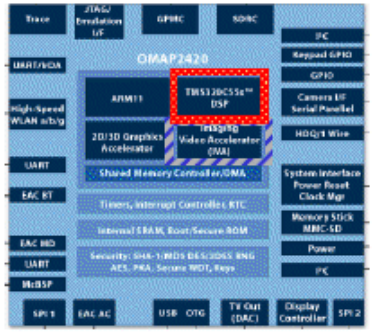
8 x SPE(DSP) + CPU
Higher clock

EMMA (NECEL)



HW Engine + DSP + CPU

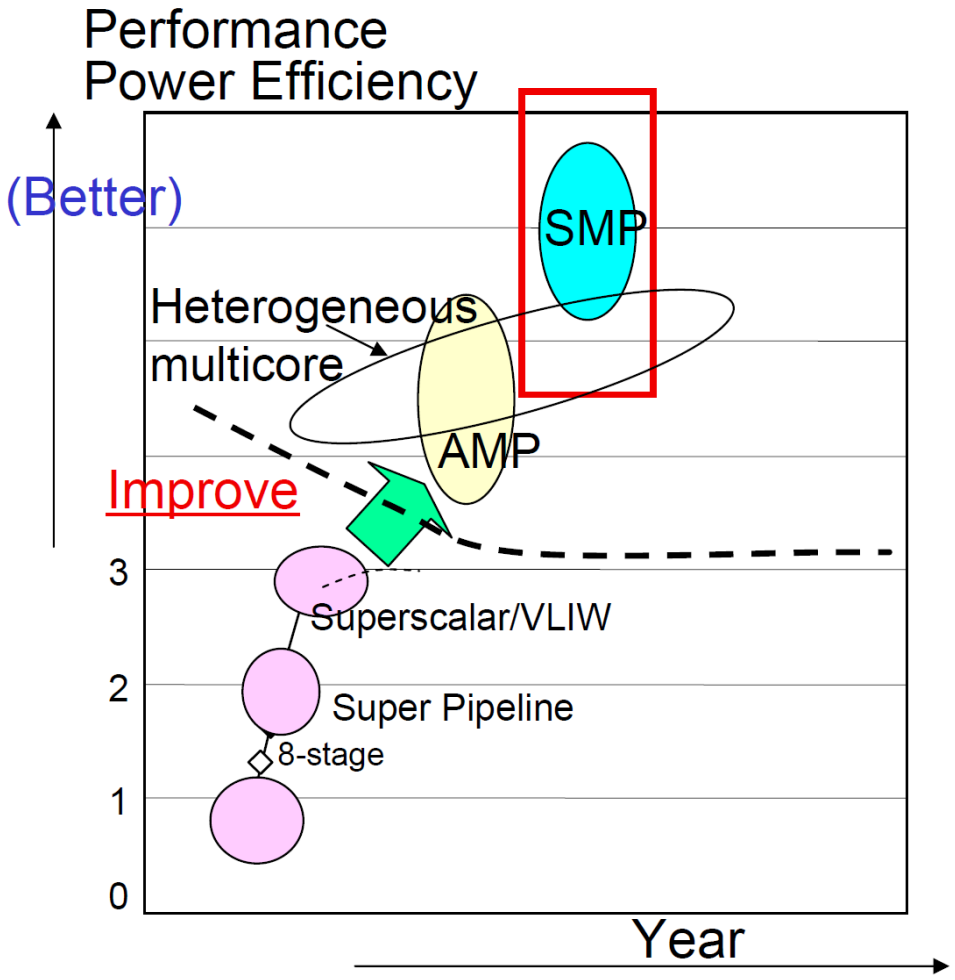
OMAP (TI)

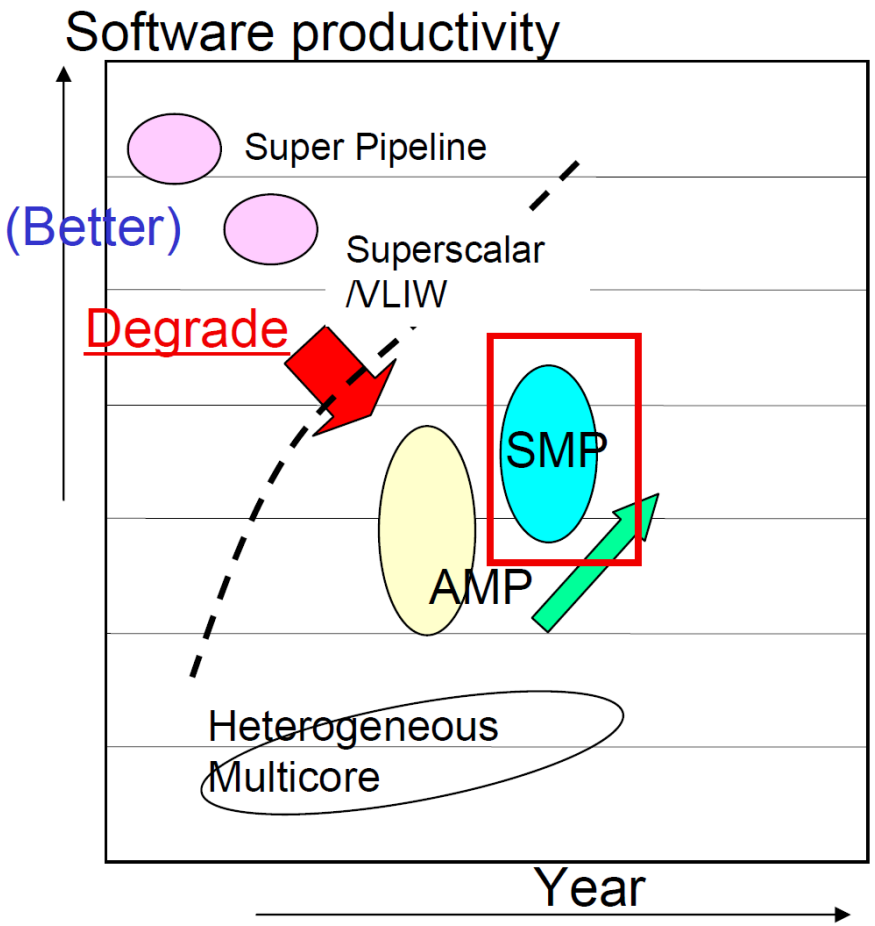


(DSP or HW) + CPU

Technology Trends for CPU Parallel Multi-core

- Multiprocessing for performance and power efficiency
- Recovery from degradation of software productivity

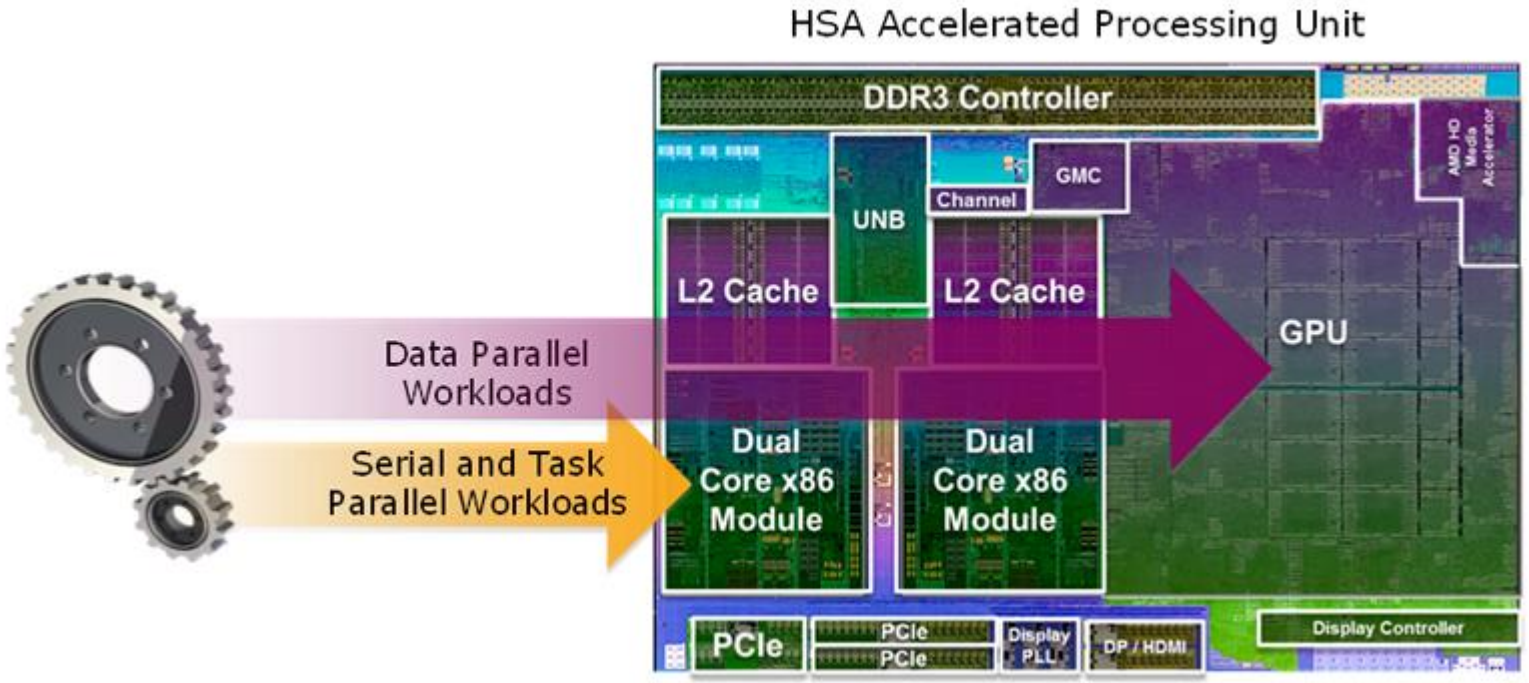




AMP/SMP: Asymmetric/Symmetric Multiprocessor

Trend: Heterogeneous System Architecture (HSA)

- Target: power, performance, programmability and portability.



A NEW ERA OF PROCESSOR PERFORMANCE

Single-Core Era

Enabled by:
- ✓ Moore's Law
- ✓ Voltage Scaling

Constrained by:
- ✗ Power
- ✗ Complexity

Assembly ➡ C/C++ ➡ Java …

Multi-Core Era

Enabled by:
- ✓ Moore's Law
- ✓ SMP architecture

Constrained by:
- ✗ Power
- ✗ Parallel SW
- ✗ Scalability

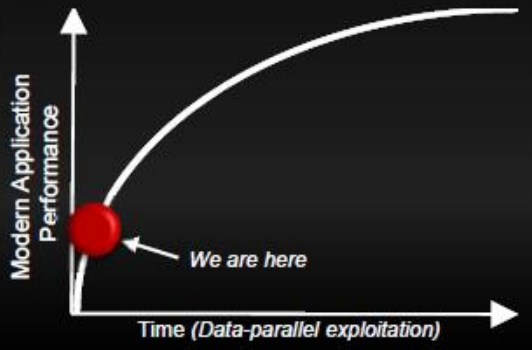
pthreads ➡ OpenMP / TBB …

Heterogeneous Systems Era

Enabled by:
- ✓ Abundant data parallelism
- ✓ Power efficient GPUs

Temporarily
Constrained by:
- ✗ Programming models
- ✗ Comm.overhead

Shader ➡ CUDA™➡ OpenCL™➡ !!!



5 | 2012 Financial Analyst Day | February 2, 2012 | Consumerization, Cloud, Convergence

AMD

EVOLUTION OF HETEROGENEOUS COMPUTING

Architecture Maturity & Programmer Accessibility

Excellent

Poor

Proprietary Drivers Era

Graphics & Proprietary Driver-based APIs

- "Adventurous" programmers
- Exploit early programmable "shader cores" in the GPU
- Make your program look like "graphics" to the GPU
- CUDA™, Brook+, etc

Standards Drivers Era

OpenCL™, DirectCompute Driver-based APIs

- Expert programmers
- C and C++ subsets
- Compute centric APIs , data types
- Multiple address spaces with explicit data movement
- Specialized work queue based structures
- Kernel mode dispatch

Architected Era

Heterogeneous System Architecture GPU Peer Processor

- Mainstream programmers
- Full C++
- GPU as a co-processor
- Unified coherent address space
- Task parallel runtimes
- Nested Data Parallel programs
- User mode dispatch
- Pre-emption and context switching

2002 - 2008 | 2009 - 2011 | 2012 - 2020

AMD

HETEROGENEOUS SYSTEM ARCHITECTURE – AN OPEN PLATFORM

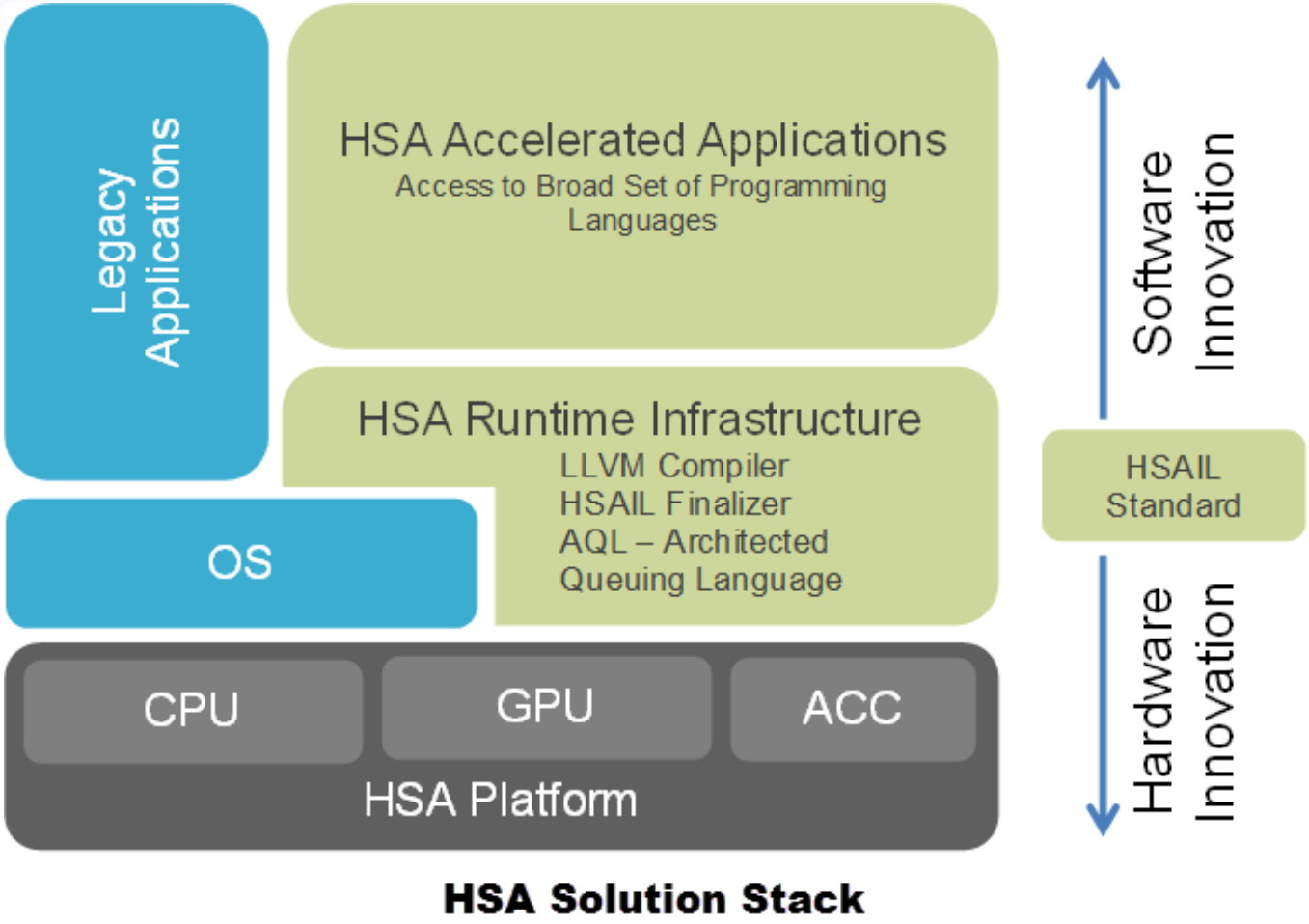
- Open Architecture, published specifications
  - HSAIL virtual ISA
  - HSA memory model
  - HSA system specification
- ISA agnostic for both CPU and GPU
- Inviting partners to join us, in all areas
  - Hardware companies
  - Operating Systems
  - Tools and Middleware
  - Applications
- HSA Foundation to guide the architecture

AMD


AMD VISION FOR HSA
Biometric Recognition:
Secure, fast, accurate: face, voice, fingerprints
Nearest Available Screen:
Push of the button enables wireless display extension
User Generated 3D Content:
New artistic uses of mainstream 3D content for everyday consumption
Augmented Reality:
Superimpose graphics, audio and other digital information as a virtual overlay
Multi-point HD Video Conferencing:
Flawless HD video anywhere, multi-stream encode.
Beyond HD Video Experiences:
Streaming media, new codecs, 3D, transcode, Audio
Natural UI & Gestures:
Touch, no touch & voice
All Day Computing:
All day active use of your PC with no need of a power source.
17 | 2012 Financial Analyst Day | February 2, 2012 | Consumerization, Cloud, Convergence
AMD

Key Founders of HSA Foundation

HSA Solution Stack



HSA Solution Stack

Example: New ARM Architecture