ECE 498: Design for the Internet of Things

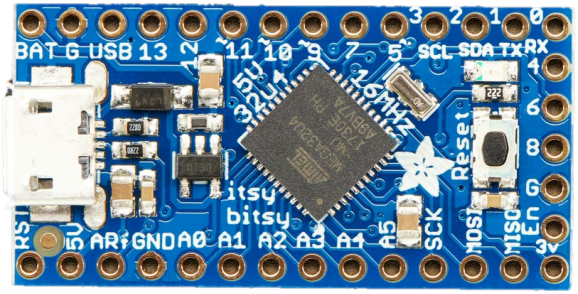
Dr. Jason Forsyth

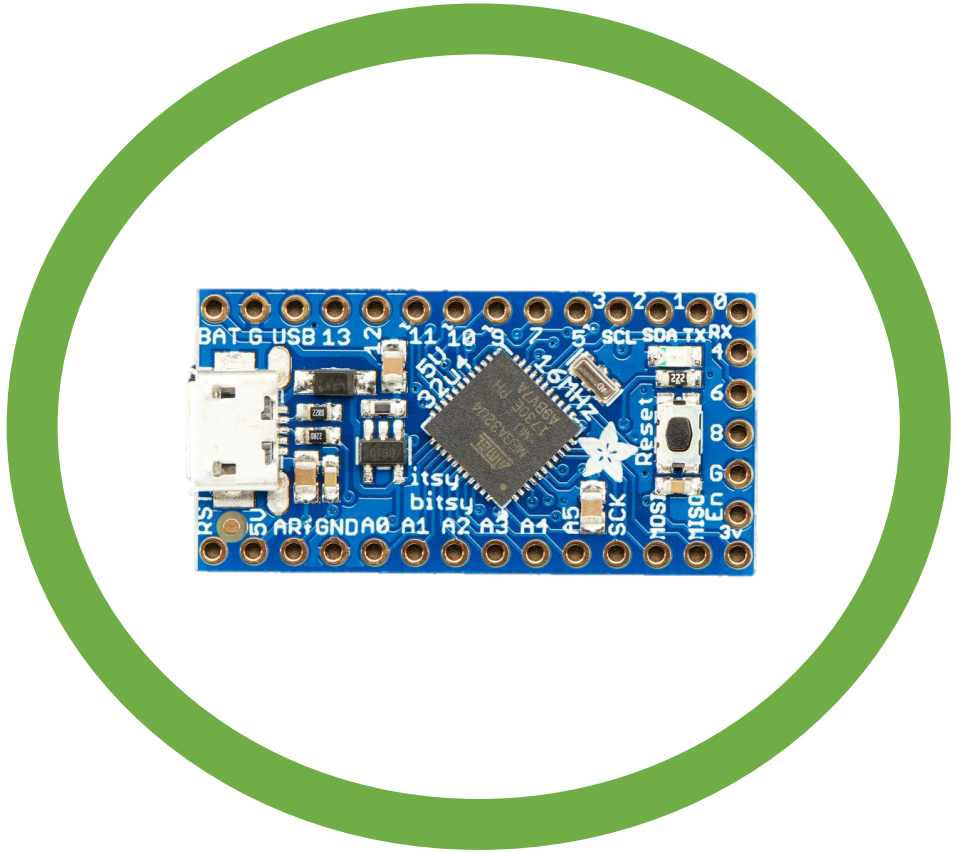
Department of Engineering

James Madison University

What kind of systems are we studying?

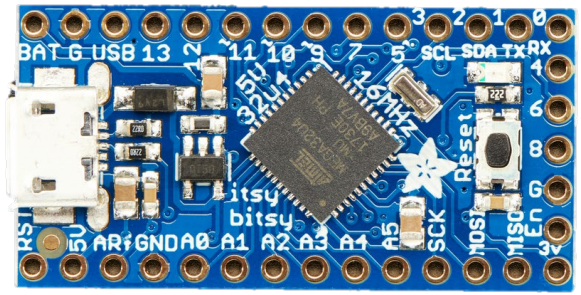
- Embedded computing system
  - Any device that includes a programmable computer but is not itself a general-purpose computer
- Limited resources (CPU, RAM, FP, Battery…etc)
- Optimize software to take advantage of hardware
- Real-time constraints; deterministic performance
- Constrained power budget




BAT G USB 13 12 11 10 9 7 5 SCL SDA TX RX
32U4
16MHz
Reset
bitsy
bitsy
RST 5V AR GND A0 A1 A2 A3 A1 A5 SCK MOSI MISO 3V


CORE™ i7
4.20GHZ

Imagine you have a choice of three processors…

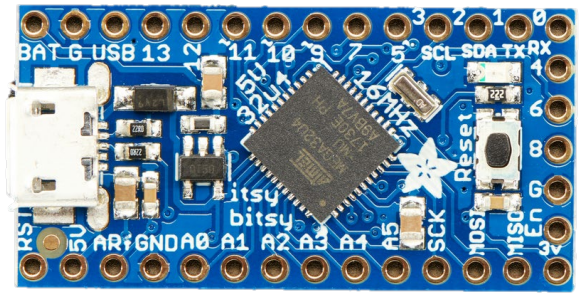


Adafruit Itsy / Arduino Pro

Silicon Labs Giant Gecko

Silicon Labs Sleepy Bee

Imagine you have a choice of three processors…



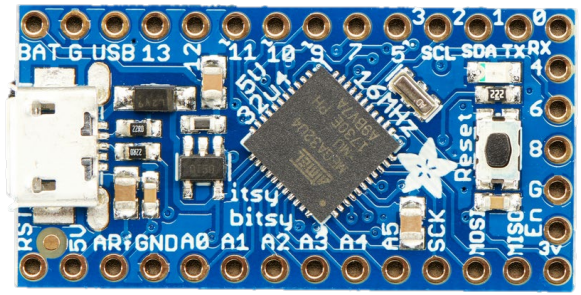
Adafruit Itsy / Arduino Pro

Silicon Labs Giant Gecko

Silicon Labs Sleepy Bee

You want to build a Oculus Rift. Which do you select?

Imagine you have a choice of three processors…



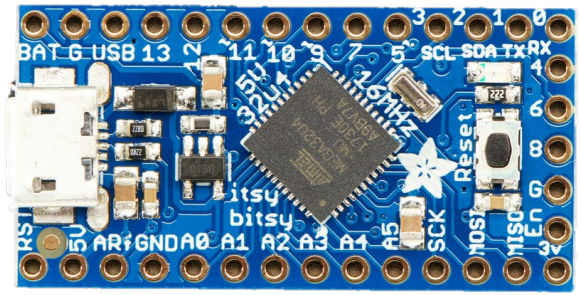
Adafruit Itsy / Arduino Pro

Silicon Labs Giant Gecko

Silicon Labs Sleepy Bee

You want to build a Bit Coin Miner. Which do you select?

Imagine you have a choice of three processors…



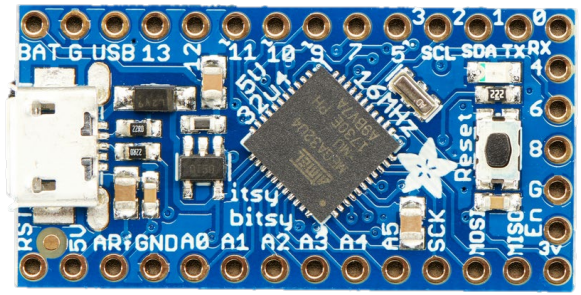
Adafruit Itsy / Arduino Pro

Silicon Labs Giant Gecko

Silicon Labs Sleepy Bee

You want to build a FitBit clone. Which do you select?

Imagine you have a choice of three processors…



Adafruit Itsy / Arduino Pro

Silicon Labs Giant Gecko

Silicon Labs Sleepy Bee

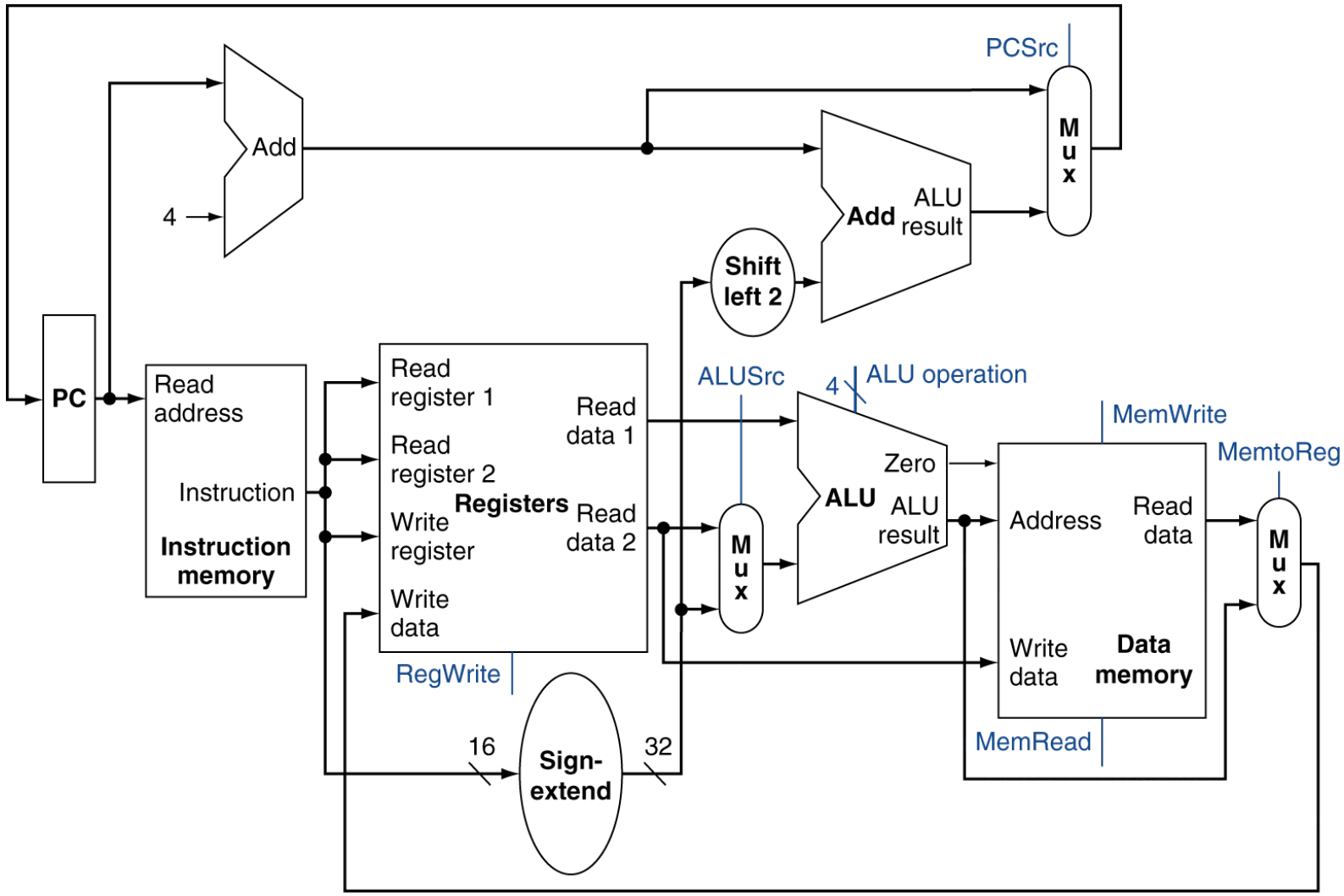
You want to build a Automotive Automated Braking System. Which do you select?

The answer is “it depends”

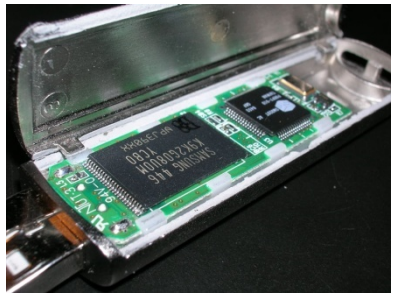
Resolving that question is the focus of this course

What aspects should be considered when selecting a processor?

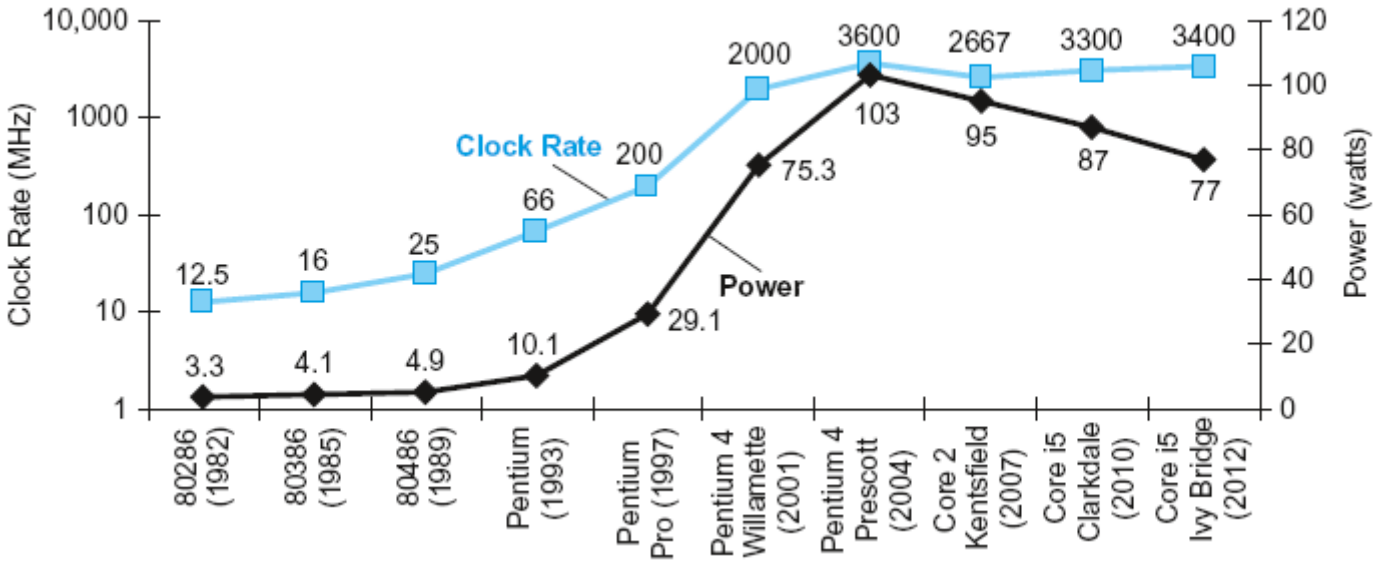
How complex is the internal architecture? What operations does it provide?



What memories are available? How much storage is there? How fast are they?



What will be the energy consumption during operation and sleep?



$$\text{Power} = \text{Capacitive load} \times \text{Voltage}^2 \times \text{Frequency}$$

×30

5V → 1V

×1000

Important Processor Considerations (my list)

- CPU Speed
- Instruction and Data word size
- Available peripherals (GPIO, ADC/DAC,UART, SPI, i2C…etc.)
- RAM, Flash, and EEPROM size and access times
- Energy efficiency
- $$$ Cost
- Tool Support / Documentation
- Corporate inertia

Why these elements are important

- CPU speed -> faster clock cycle means more instructions per second
  - Recall CPI = cycles / instruction
- More Flash -> more space for programs
  - Your PC programs are MB in size. You may only have several KB.
- More RAM -> deeper stack and heap. More “storage”.
  - Think of how efficient you will need to code if RAM is 1Kbits
- More EEPROM -> more long-term storage for data
  - Where do you store your data after a power down?
- More peripherals -> able to communication with more devices
- Energy efficiency -> do you have sufficient battery life to do the job?

Things that are harder to determine

- Data word size
  - Larger data size -> can operate on ”larger” variables, but does the application really need all 32 bits? Can we get away with 8bits?
  - *floats*, *longs*, and *doubles* spread across smaller word sizes

- Instruction word size
  - Larger instructions may be more “complex” and can likely perform multiple operations at once.
  - Smaller word size means smaller program, but maybe it requires more operations

When studying Microprocessors, the answer to everything is, “it depends”

- There are hundreds of different processors from all generations
  - SleepyBee is a modern IBM 8051 clone, circa 1980 (link)
  - Arduino Uno is an Atmel ATmega, circa 1996 (link)
  - Giant Gecko contains an ARM Cortex-M3, circa 2004 (link)

- How do you compare/select devices separated by two decades?

- **One outcome of this course is to help you determine what is the right solution**

What happens when we get it wrong?

Toyota to Pay $1.2B for Hiding Deadly 'Unintended Acceleration'

By BRIAN ROSS, JOSEPH RHEE, ANGELA M. HILL, MEGAN CHUCHMACH and AARON KATERSKY ·
March 19, 2014

Toyota Motor Corp. vehicles sit parked ahead of shipment outside the Central Motor Corp. plant in Ohira, Miyagi Prefecture, Japan, March 7, 2014.

Car manufacturer Toyota has agreed to pay a staggering $1.2 billion to avoid prosecution for covering up severe safety problems with "unintended acceleration," according to court documents, and continuing to make cars with parts the FBI said Toyota "knew were deadly."

A deferred prosecution agreement, filed today, forced Toyota to "admit" that it "misled U.S. consumers by concealing and making deceptive statements about two safety related issues affecting its vehicles, each of which caused a type of unintended acceleration."

Toyota "put sales over safety and profit over principle," according to FBI Assistant Director George Venizelos.

With hacking, music can take control of your car

Remote-controlled car hacking is a real possibility, researchers say

By Robert McMillan

IDG News Service | March 14, 2011

RELATED TOPICS

Security Business

About 300 years ago, the English playwright William Congreve wrote, "music has charms to soothe a savage breast, to soften rocks, or bend a knotted oak." This week we learned that it can also help hackers break into your car.

Researchers at the University of California, San Diego, and the University of Washington have spent the past two years combing through the myriad computer systems in late-model cars, looking for security flaws and developing ways to misuse them. In a new paper, they say they've identified a handful of ways a hacker could break into a car, including attacks over the car's Bluetooth and cellular network systems, or through malicious software in the diagnostic tools used in automotive repair shops.

But their most interesting attack focused on the car stereo. By adding extra code to a digital music file, they were able to turn a song burned to CD into a Trojan horse. When played on the car's stereo, this song could alter the firmware of the

MORE GOOD READS

Car-hacking: Bluetooth and other security issues

Car hackers can kill brakes, engine, and more

Car tech: Electric vehicles get an IT assist

on IDG Answers

I'm considering a slight career change to IT security - what do I need to...

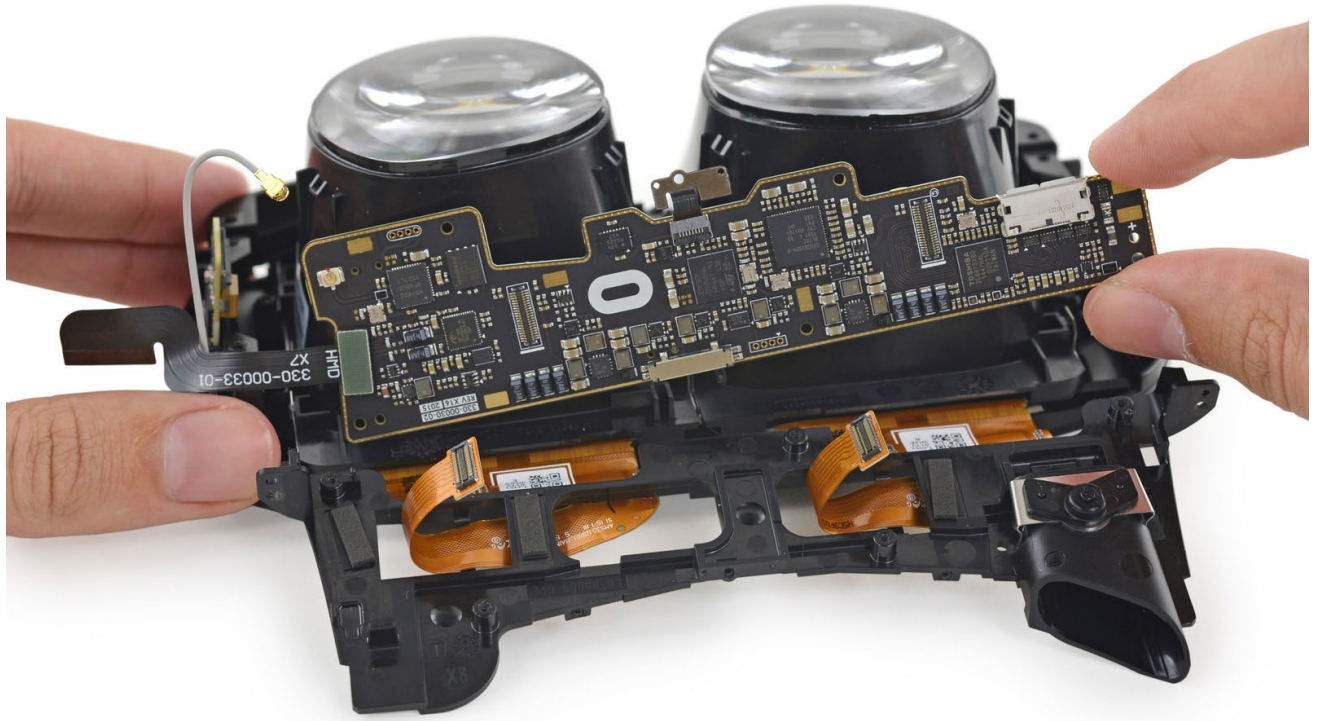
Save $100 on Xbox One Console - Limited Edition Halo 5: Guardians Bundle -...

“Bad” Microprocessors in the News

- Toyota Prius
  - 100M lines of code; 30 microprocessors across the car
  - One bad line of code, missed deadline, or random bit fluctuation -> sudden acceleration in the break control. “Task X death”
  - 10,000 global variables (link, slide 40)

- Ford / MP3 Hack (link)
  - Audio system and ECU on same CAN bus

Going forward, what happens if you blow up the stack or use a null pointer?



Tesla car that crashed and killed driver was running on Autopilot, firm says

- **Company says driver took no action despite system's warnings**
- **Uber settles with family of woman killed by self-driving car**

▲ Emergency personnel work at the scene where a Tesla electric SUV crashed into a barrier on US Highway 101 in Mountain View, California. Photograph: AP

Tesla has said a car that crashed in California last week, killing its driver, was operating on Autopilot.

The 23 March crash on highway 101 in Mountain View is the latest accident to involve self-driving technology. Earlier this month, a self-driving Volvo SUV that was being tested by the ride-hailing service Uber struck and killed a pedestrian in Arizona.

Why the distinction between “Embedded” and ”Desktop” computing matters…

- Embedded systems have real-time deadlines

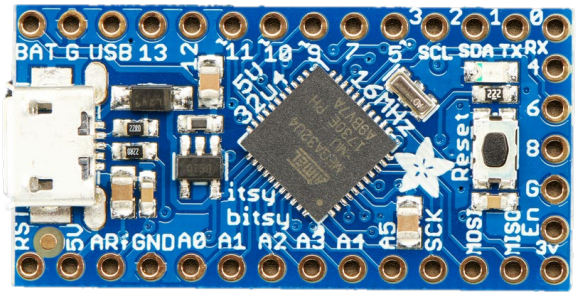
- Hard-deadline: missed deadline will cause system failure
  - Example: failure to detect collision with object and/or apply breaks quickly

- Soft-deadline: missed deadline will reduce system quality but not cause failure
  - Example: low bandwidth causes blurry Facetime conversation
  - Example: AppleTV doesn’t wake-up fast enough. Delaying Game of Thrones viewing

Why the distinction between “Embedded” and ”Desktop” computing matters…

- To meet real-time deadlines, the system must be deterministic.

- 4 cores / 8 threads executing independently
- Instruction execution is *speculative* and *out of order*
- Context switches from OS
- Unknown memory access times due to cache and RAM

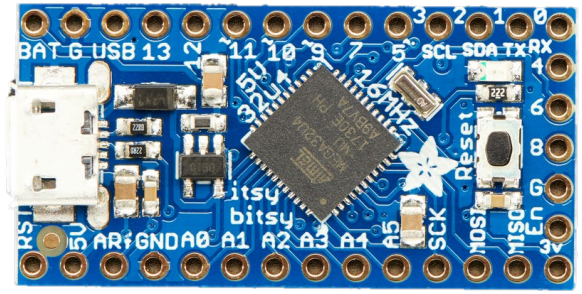


- 1 core / 1 thread but with interrupts
- Instruction execution is in-order; 1 CPI
- Supports simple Real-Time OS (RTOS)
- Constant memory access times; no cache

Why working with Embedded Systems is challenging

- Embedded systems are <u>resource constrained</u>

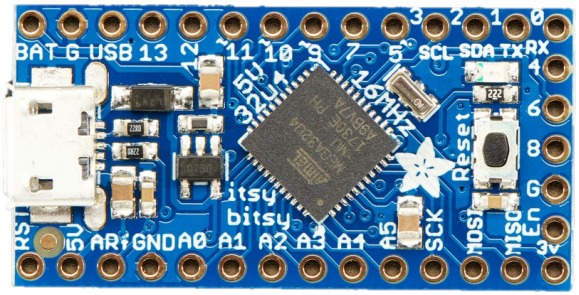
- 3 GHz
- 64 GB RAM
- SSD hard drive
- Connected to wall outlet!
- Floating point / Graphics units



- ~16MHz
- KiloBytes of RAM
- What hard drive?
- Run on battery for weeks
- Floating multiply?

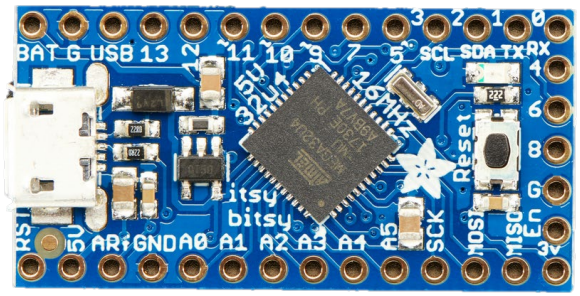
Why working with Embedded Systems is challenging

- Embedded systems have a limited power budget



$$P = \underbrace{\alpha C V^2 f}_{\text{Dynamic Power}} + \underbrace{V I_{leak} + \tau \alpha V I}_{\text{Static Power}}$$

In-class Activity: Comparison of Various Microprocessors



Arduino Uno

Silicon Labs Giant Gecko

Silicon Labs Sleepy Bee

Feature	Atmel 32u4	Giant Gecko	Sleepy Bee
Processor Model Number	ATmega32u4	EFM32-GG990F1024 (ARM Cortex-M3)	EFM8SB-10F8G-A-QFN24 (CIP-51 8051 core)
Typical Clock Speed (MHz)			
Data Word Size			
RAM (KB)			
Flash Size (KB)			
EEPROM (KB)			
Operating Voltage Range (V)			
Peripherals			
Cost (Processor Only)			

Feature	Atmel 32u4	Giant Gecko	Sleepy Bee
Processor Model Number	ATmega32u4	EFM32-GG990F1024 (ARM Cortex-M3)	EFM8SB-10F8G-A-QFN24 (CIP-51 8051 core)
Typical Clock Speed (MHz)	8MHz @3.3V / 16 MHz @ 5V	48 MHz	25 MHZ
Data Word Size	8-bits	32-bits	8-bits
RAM (KB)	2.5KB	128 KB	0.5 KB (512 bytes) 64 words
Flash Size (KB)	32KB	1024 KB	8 KB
EEPROM (KB)	1KB	0 KB	0 KB
Operating Voltage Range (V)	2.7 – 5.5V	1.98-3.8V	1.8-3.6V
Peripherals	Timers, RTC, PWM, ADC, DAC, USART, SPI, i2C, WDT	All of Arduino + LCD, AES, Op-Amps, USB, Energy monitor	All of Arduino + CRC check, capacitive sense
Cost (Processor Only)	$3.50	$8	$0.93

Feature	Atmel 32u4	Giant Gecko	Sleepy Bee
Processor Model Number	ATmega32u4	EFM32-GG990F1024 (ARM Cortex-M3)	EFM8SB-10F8G-A-QFN24 (CIP-51 8051 core)
Typical Clock Speed (MHz)	8MHz @3.3V / 16 MHz @ 5V	48 MHz	25 MHZ
Harvard / Von Neumann	Harvard (separate bus)	Harvard (separate bus)	Harvard (separate bus)
Estimated MIPS	8-16 MIPS	1.25 MIPS/MHz * 48 MHz = 60 MIPS	25 MIPS
Data Word Size	8-bits	32-bits	8-bits
RAM (KB)	2.5KB	128 KB	0.5 KB (512 bytes) 64 words
Flash Size (KB)	32KB	1024 KB	8 KB
EEPROM (KB)	1KB	0 KB	0 KB
Operating Voltage Range (V)	2.7 – 5.5V	1.98-3.8V	1.8-3.6V
Peripherals	Timers, RTC, PWM, ADC, DAC, USART, SPI, i2C, WDT	All of Arduino + LCD, AES, Op-Amps, USB, Energy monitor	All of Arduino + CRC check, capacitive sense
Cost (Processor Only)	$3.50	$8	$0.93

Comparison of Processors

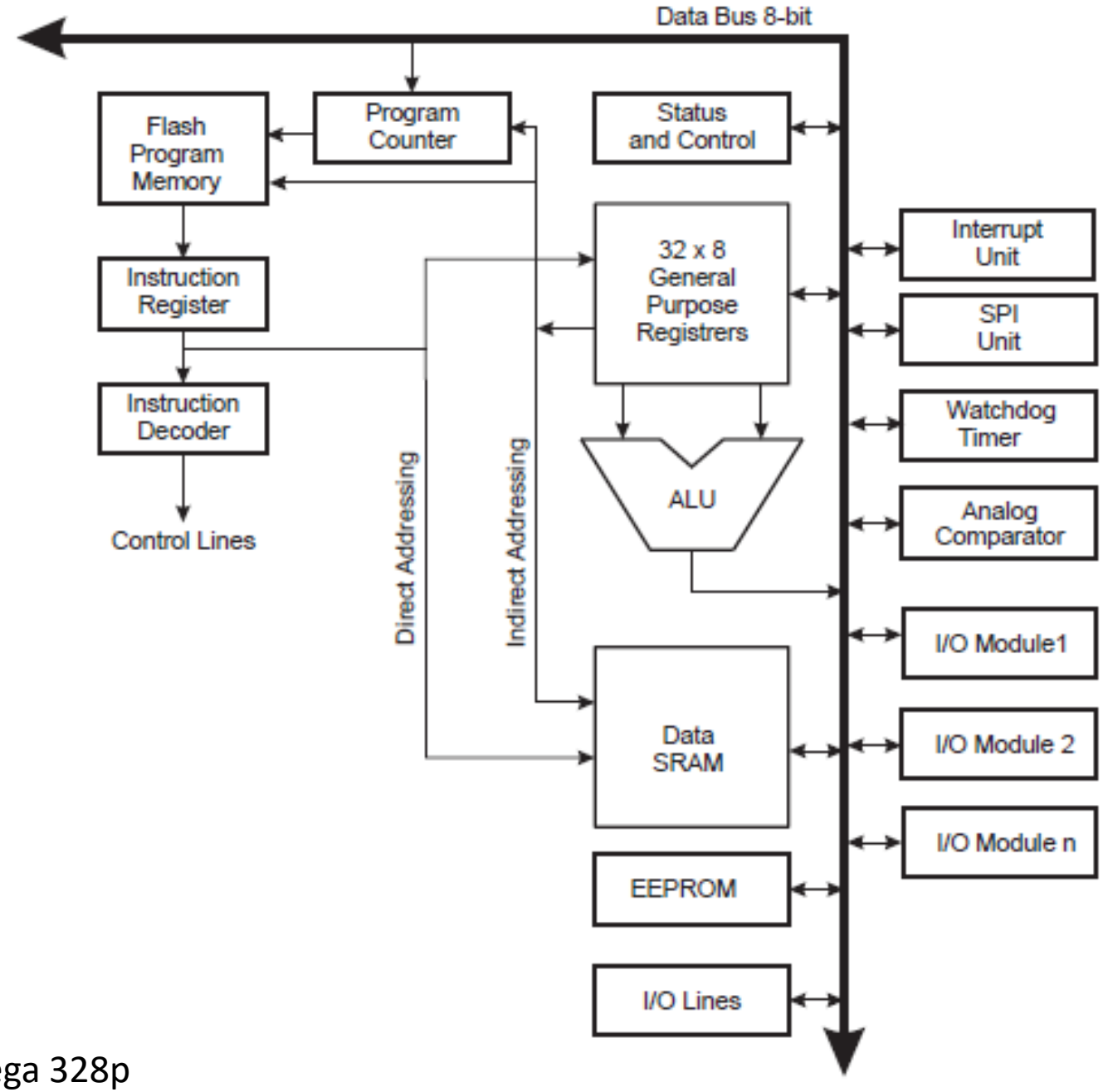
- Arduino and Sleepy Bee are similar: price point, peripherals, memory…etc.

- Giant Gecko has more performance @ 48MHz and significantly more peripherals
  - Likely could run Linux or other basic kernel
  - Sleepy Bee barely runs modified C

- However there are significantly different than the processors you are used to….

Feature	Arduino Uno	Giant Gecko	Sleepy Bee	Intel i7
Processor	ATmega32u4	EFM32-GG990F1024 (ARM Cortex-M3)	EFM8SB-10F8G-A-QFN24 (CIP-51 8051 core)	i7-6700HQ (link)
Typical Clock Speed	8MHz @3.3V / 16 MHz @ 5V	48 MHz	25 MHZ	2.6 GHz (3.5 GHz turbo)
Harvard / Von Neumann	Harvard (separate bus)	Harvard (separate bus)	Harvard (separate bus)	Von-Neumann (unified program/data)
Estimated MIPS	8-16 MIPS	1.25 MIPS/MHz * 48 MHz = 60 MIPS	25 MIPS	304,510 MIPS (link)
Data Word Size	8-bits	32-bit	8-bit / ?	64-bit
RAM	2.5KB	128 KB	0.5 KB (512 bytes)	64 GB
Flash Size	32KB	1024 KB	8 KB	0 KB
EEPROM	1KB	0 KB	0 KB	0 KB
Typ. Voltage Range	2.7 – 5.5V	1.98-3.8V	1.8-3.6V	~1.2V (45W)
Peripherals	Timers, RTC, PWM, ADC, DAC, USART, SPI, i2C, WDT	All of Arduino + LCD, AES, Op-Amps, USB, Energy monitor	All of Arduino + CRC check, capacitive sense	Too many to list
Cost	$3.50	$8	$0.93	$378.00

Architectural Differences



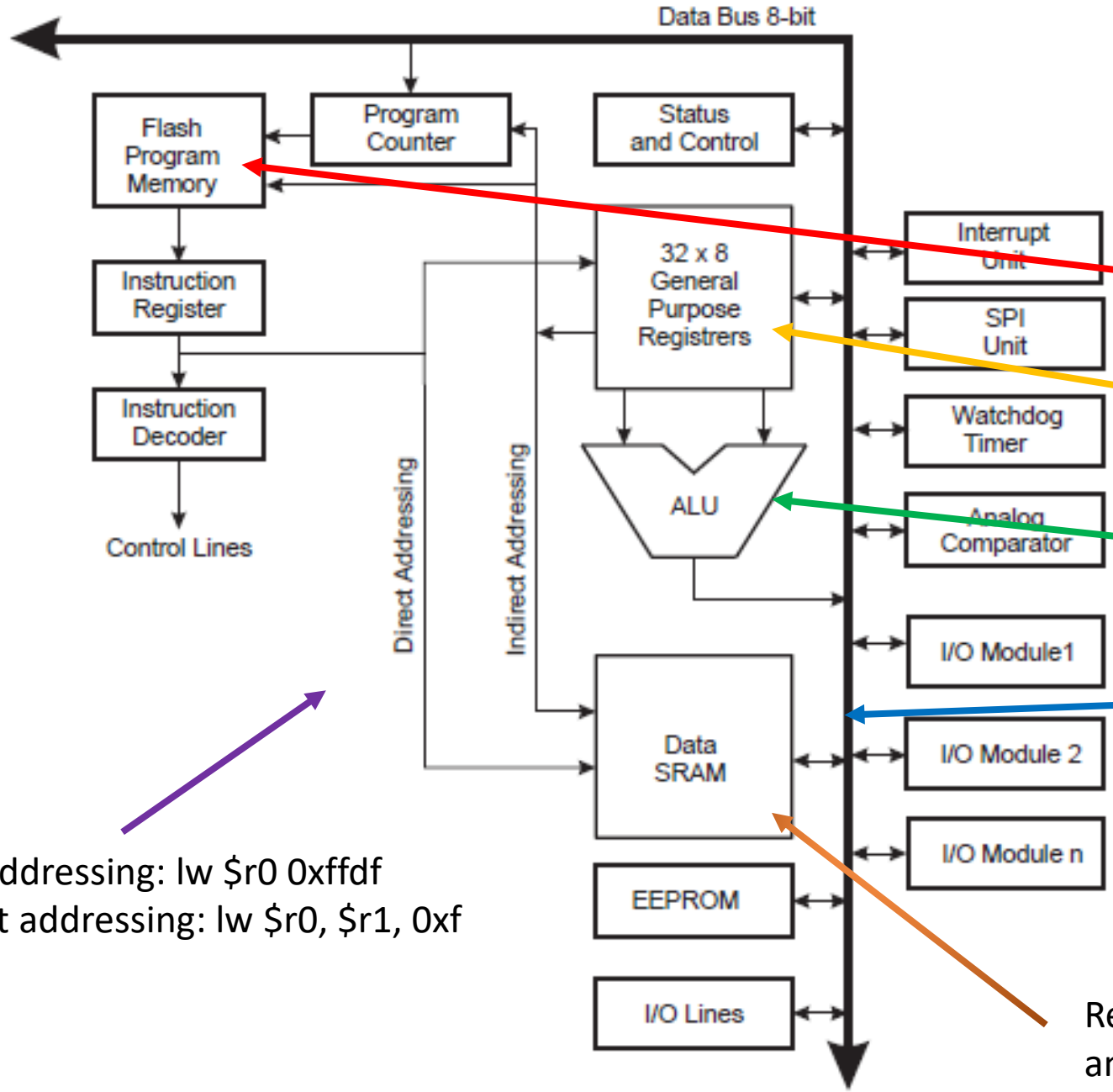
Where does your program live?

Where are your registers?

Where are instructions executed?

Where are peripherals attached?

ATmega 328p



Where does your program live?

Where are your registers?

Where are instructions executed?

Where are peripherals attached?

Direct addressing: lw $r0 0xffdf
In-direct addressing: lw $r0, $r1, 0xf

Recall the difference between SRAM and DRAM…

ATMega System Overview - Notes

In order to maximize performance and parallelism, the AVR uses a Harvard architecture – with separate memories and buses for program and data. Instructions in the program memory are executed with a single level pipelining. While one instruction is being executed, the next instruction is pre-fetched from the program memory. This concept enables instructions to be executed in every clock cycle. The program memory is In-System Reprogrammable Flash memory.

8.5 I/O Memory

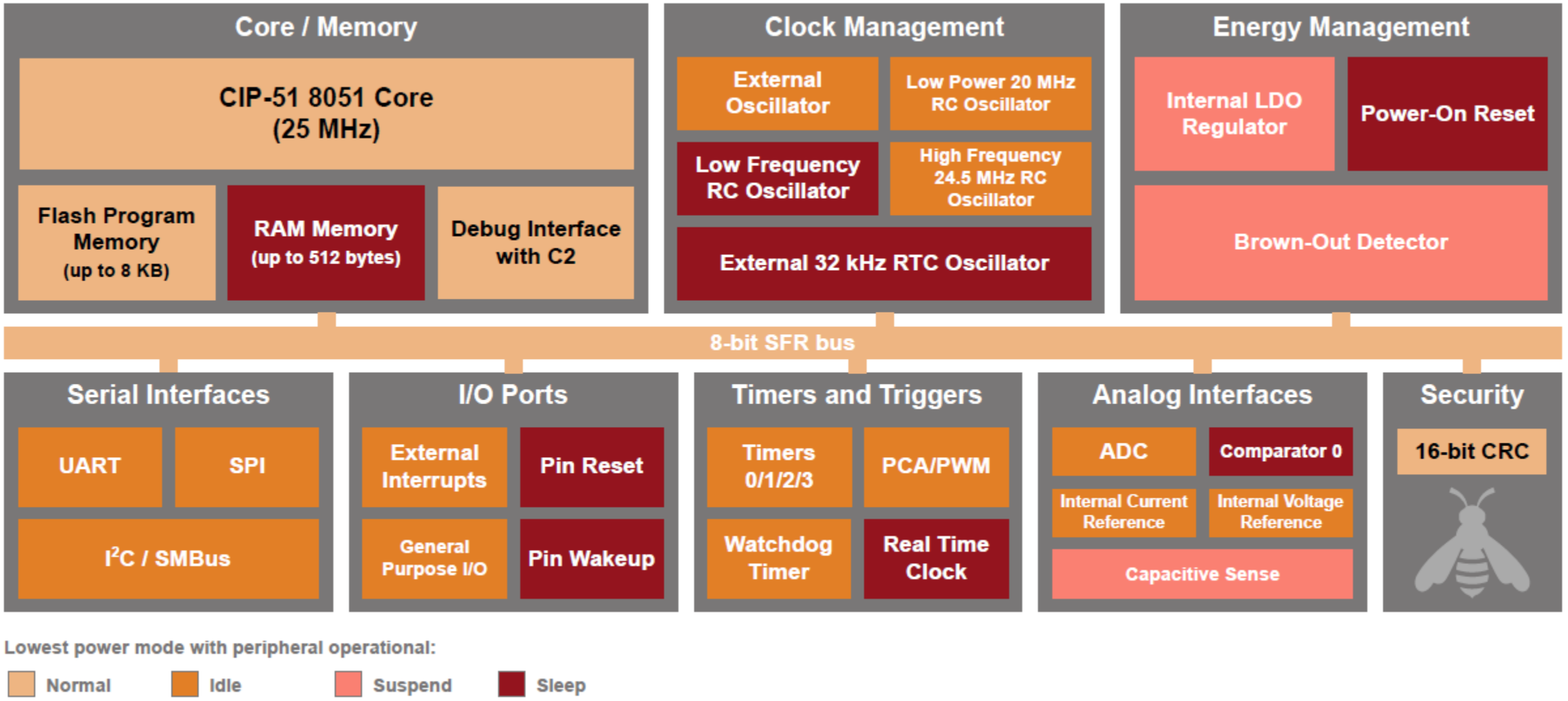
The I/O space definition of the ATmega48A/PA/88A/PA/168A/PA/328/P is shown in "Register Summary" on page 612.

All ATmega48A/PA/88A/PA/168A/PA/328/P I/Os and peripherals are placed in the I/O space. All I/O locations may be accessed by the LD/LDS/LDD and ST/STS/STD instructions, transferring data between the 32 general purpose working registers and the I/O space. I/O Registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions. Refer to the instruction set section for more details. When using the I/O specific commands IN and OUT, the I/O addresses 0x00 - 0x3F must be used. When addressing I/O Registers as data space using LD and ST instructions, 0x20 must be added to these addresses. The ATmega48A/PA/88A/PA/168A/PA/328/P is a complex microcontroller with more peripheral units than can be supported within the 64 location reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 - 0xFF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

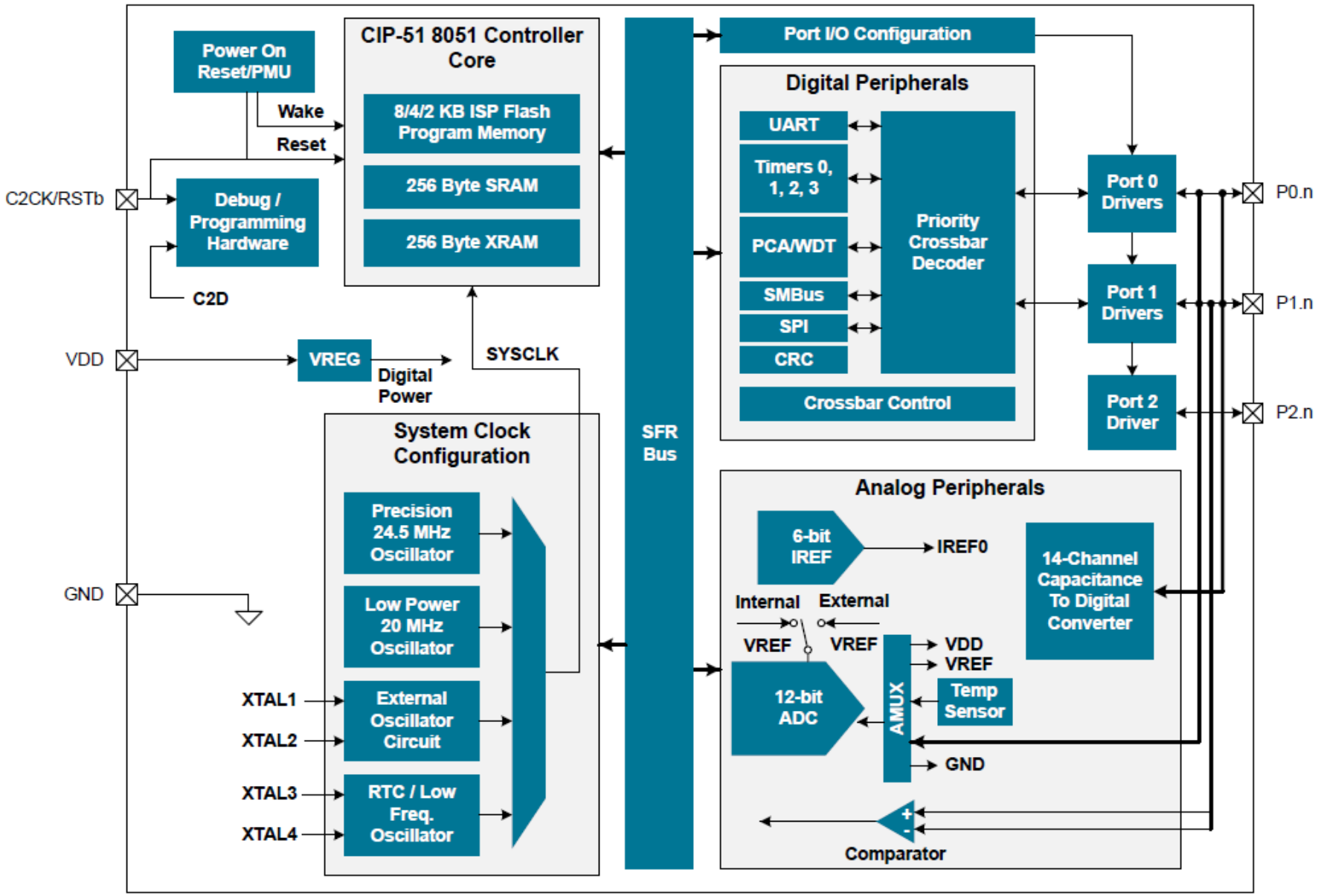
Single cycle execution

Memory mapped I/O

Sleep Bee – Consumer Diagram



Sleepy Bee – Real Diagram



Where does your program live?

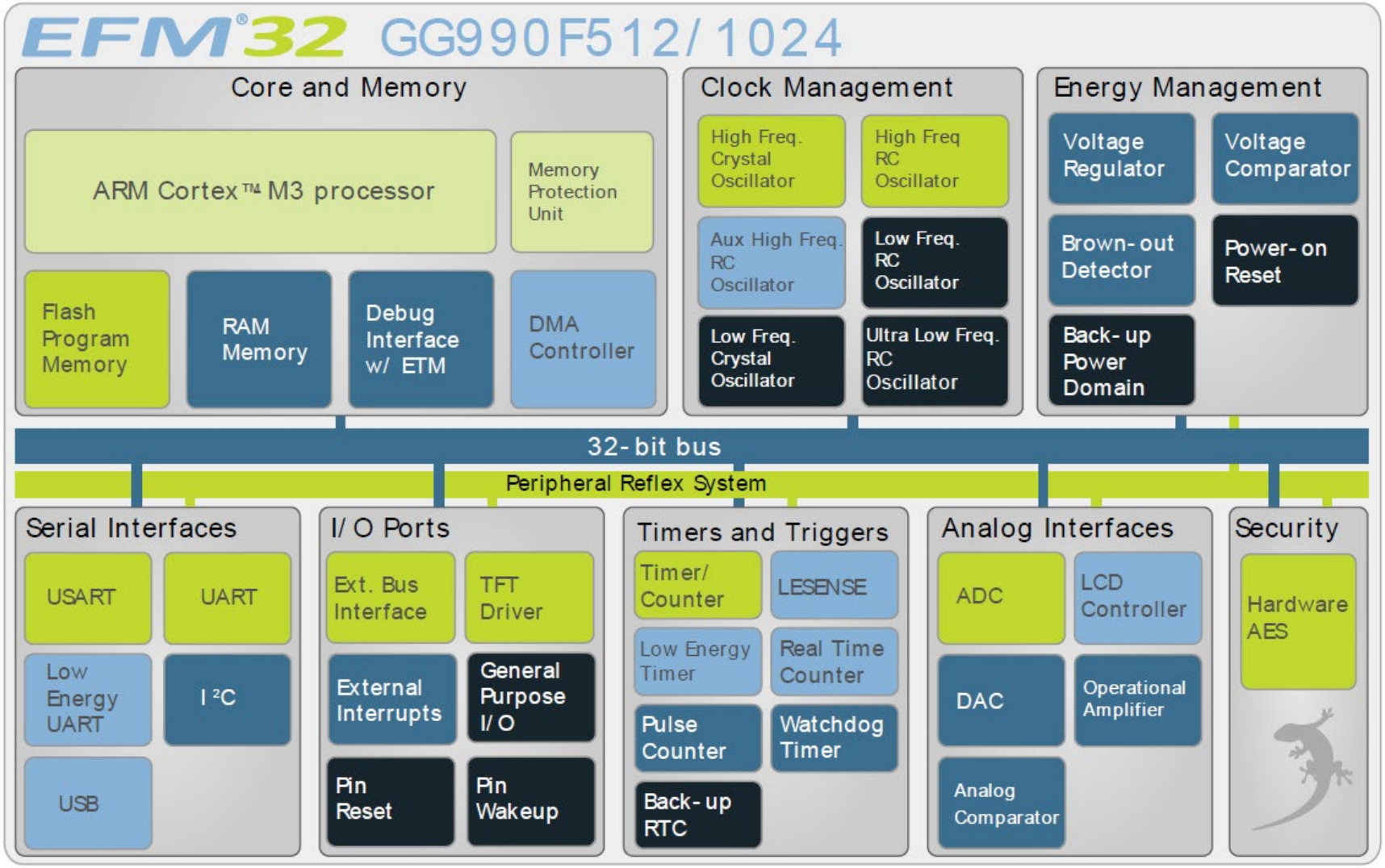
Where are your registers?

Where are instructions executed?

Where are peripherals attached?

Where do the clocks come from?

Giant Gecko – Consumer Diagram



Why choose the “dumbest” one

- The simplest system will likely be the most predictable
- Pick the smallest, cheapest, and easiest to use processor that gets the job done.
- Possible to always upgrade to a new unit in the “family”.

Processor Families

- While processors differ across manufacturers, they are likely to be very similar with their own “family”
  - Differences occur in # of GPIO, ADC, DAC, Timers…etc.
  - Flash/RAM/EEPROM size…etc.
- Coding and board design can remain similar within the family. Also, don’t have to learn a new architecture each time
- Allows development on a smaller/larger platform and then scale the system with production or demand

8-bit AVR® Microcontrollers Peripheral Integration

Quick Reference Guide

Product Family	Pin Count	Program Flash Memory (KB)	SRAM (KB)	Peripheral Function Focus																																			
				Intelligent Analog							Waveform Control				Timing and Measurements				Logic, Crypto and Math			Safety and Monitoring				Communications						User Interface			System Flexibility						
				ADC (# of bits)	ADC (# of channels)	Comparators	ADC Gain Stage	DAC (# of bits)	Temperature Sensor	Internal Voltage Reference	8-bit PWM	16-bit PWM	Quadrature Decoder	Waveform Extension	Real-Time Counter	8-bit Timer/Counters	12-bit Timer Counter	16-bit Timer/Counter	CCL	MULT	Crypto (AES/DES)	CRC	POR	BOD	WDT	UART	USART	USB	I²C	SPI	IRCOM	QTouch® Technology	QTouch Technology with PTC[2]	LCD	External Bus Interface	DMA Channels	Event System	SleepWalking	Sleep Modes	picoPower® Technology
ATtiny4/5/9/10	6	0.5–1	0.032	10[3]	4[3]	✓						2						1					✓		✓							✓						4	
ATtiny102/104	8/14	1	0.032	10	5/8	✓				✓		2						2					✓		✓		1											4	
ATtiny13A	8–20	1	0.064	10	4	✓						2											✓	✓	✓							✓						3	✓
ATtiny20/40	12–20	2/4	0.128/0.256	10	8/12	✓			✓		2	2				1		1					✓	✓	✓				1	1		✓						4	
ATtiny24A/44A/84A	14–20	2–8	Up to 0.512	10	8	✓	✓		✓	✓	2	2				1		1					✓	✓	✓				1	1		✓						4	✓
ATtiny25(V)/45(V)/85(V)	8–20	2–8	Up to 0.512	10	4	✓	✓		✓	✓	4					2							✓	✓	✓				1	1		✓						3	
ATtiny48/88	28–32	4/8	Up to 0.512	10	8	✓			✓	✓	1	1				1		1					✓	✓	✓				1	1								3	✓
ATtiny87/167	20–32	8/16	0.512	10	11	✓			✓	✓	1	2				1		1					✓	✓	✓	1[1]			1	2								4	
ATtiny261A/461A/861A	20–32	2–8	Up to 0.512	10	11	✓	✓		✓	✓						1		1					✓	✓	✓				1	1		✓						4	✓
ATtiny20x/40x/80x/160x	8–24	2–16	Up to 1	10	12	✓			✓	✓		2			✓			1	✓	✓		✓	✓	✓	✓		1[1]		1	1					✓	✓		3	✓
ATtiny21x/41x/81x/161x/321x	8–24	2–32	Up to 2	10	12	✓		8	✓	✓		2			✓		1	1	✓	✓		✓	✓	✓	✓		1[1]		1	1		✓[4]			✓	✓		3	✓
ATtiny441/841	14–20	4/8	Up to 0.512	10	12	✓	✓		✓		1	2				1		2					✓	✓	✓		2		1	1								4	✓
ATtiny1634	20	16	1	10	12	✓			✓	✓	2	2				1		1					✓	✓	✓		2		1			✓						4	✓
ATtiny2313A	20	2	0.128	–	–	✓				✓	2	2				1		1					✓	✓	✓		1		1	2								3	✓
ATmega8A/16A/32A	28–44	8–32	1–2	10	8	✓					2	1			✓	2		1		✓			✓	✓	✓		1		1	1		✓						5	
ATmega8U2/16U2/32U2	32	8–32	0.5–1	–	–	✓			✓	✓	4	6			✓	2		3		✓			✓	✓	✓		2	✓	2	2			✓					6	
ATmega16U4/32U4	32	16/32	1/2	10	12	✓			✓	✓	5					1		1		✓			✓	✓	✓		1	✓		1								6	
ATmega48PB/88PB/168PB/328PB	32	4–32	0.5–2	10	8	✓			✓	✓	4	2/6[6]			✓	2		1/3[6]		✓			✓	✓	✓		1/2[6]		1/2[6]	1/2[6]		✓	1[6]					6	
ATmega320x/480x	28–48	32–48	Up to 6	10	16	✓			✓	✓	4	3			1			5	✓	✓		✓	✓	✓	✓	4			1	1							✓		3	✓
ATmega64A/128A	64	64–128	4	10	8	✓	✓			✓	2	6				2		2		✓			✓	✓	✓		2		1	1		✓						6	

Summary

- Embedded systems are significantly different than traditional PCs
  - Real-time deadlines; power budget; resource constraints; deterministic operation
- Microprocessors exhibit a wide variations across manufacturers; however they are similar within “families”
  - Chips in the same family share the same core/architecture but may differ in memory and peripheral resources
- Selecting a processor for a particular application is challenging.
  - There are many parameters to consider, some are non-technical in nature