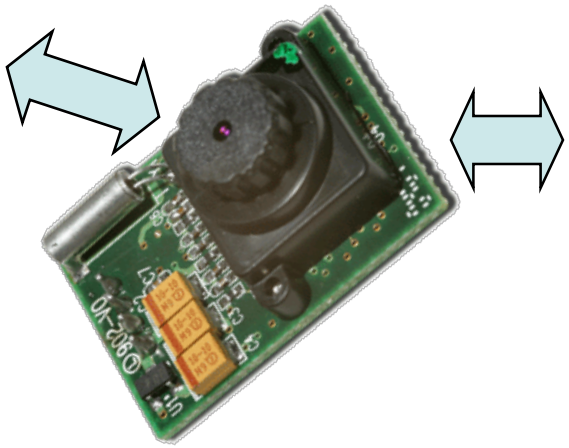
ECE 332 – Embedded Systems Lab



```
int circ_buffer[N], circ_buffer_head = 0;
int c[N]; /* coefficients */
…
int ibuf, ic;
for (f=0, ibuff=circ_buff_head, ic=0;
      ic<N; ibuff=(ibuff==N-1?0:ibuff++),
      ic++)
f = f + c[ic]*circ_buffer[ibuf];
```

Acknowledgement

- These slides are a result of cumulative contributions from Prof. Burleson, Koren, Kundu and Moritz.

- Materials have also been adopted from the textbook: *Wolf, Computers as Components, Morgan Kaufman*, 2005

Outline for Today's Lecture

- Feedback on Lab1
- Embedded CPUs
- Interrupts
- IO
- Embedded System Software Design
- Basics on Image Manipulation

Readings

- Chapters 2, 3
  - CPUs, Interrupts
- Chapter 4, emphasizing 4.1 through 4.7
  - I/O, USB, debugging
- Chapter 5, emphasizing 5.1-5.4, 5.9
  - Programming, patterns, test …

Objectives of Lab 2

- Sense the external world through a camera
- Transfer the camera data to the DE1-SoC board
- Process the image in software on the ARM CPU
- View the images on the monitor

- Concepts:
  - I/O (camera, monitor, interrupts)
  - Basic image processing
  - Embedded software design and test running on an embedded processor

Outline for Today's Lecture

- Administrative, Lab2, Feedback on Lab1
- Embedded CPUs
- Interrupts
- IO
- Embedded System Software Design
- Basics on Image Manipulation

RISC vs Superscalar

- RISC pipeline executes one instruction per clock cycle (usually).
  - For example, ARM, MIPS, PowerPC, etc
- Superscalar machines execute multiple instructions per clock cycle.
  - Faster execution.
  - More variability in execution times.
  - More expensive CPU.
  - Requires a lot of hardware.
    - n2 instruction unit hardware for n-instruction parallelism.
  - For example, Intel X86.

Order of execution

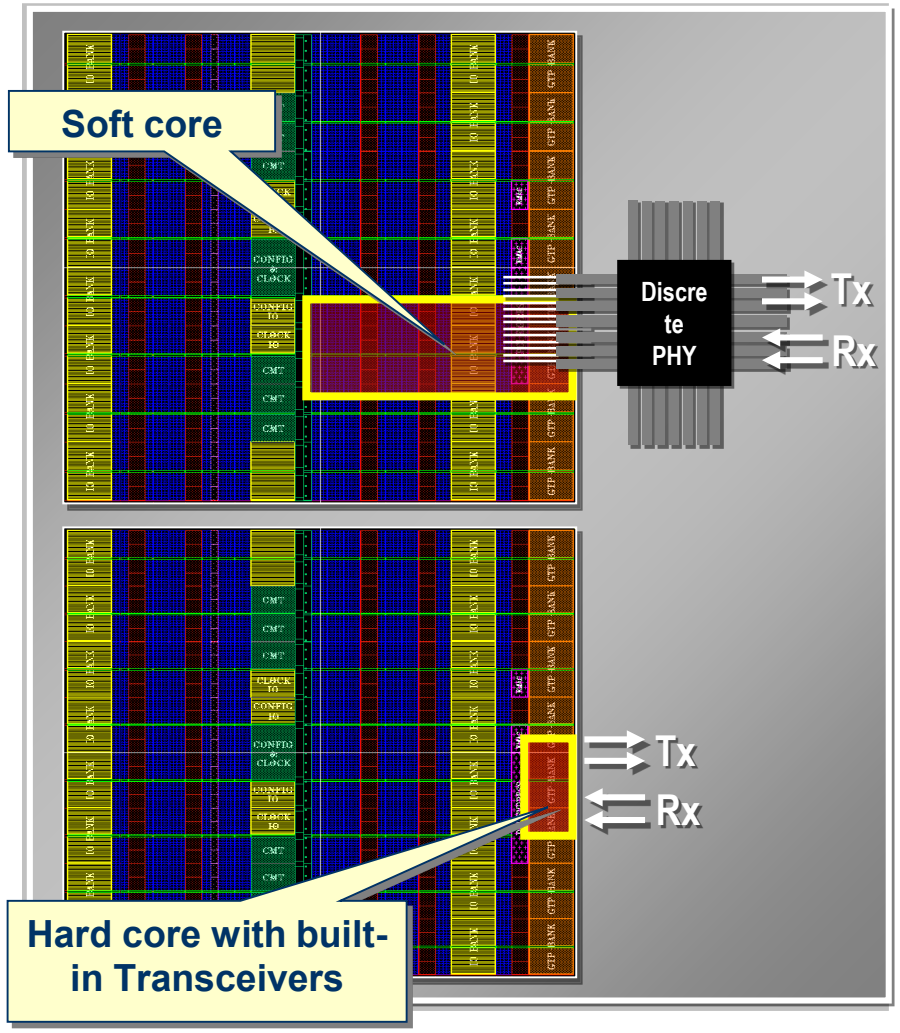
- In-order:
  - Machine stops issuing instructions when the next instruction can’t be dispatched.
- Out-of-order:
  - Machine will change order of instructions to keep dispatching.
  - Substantially faster but also more complex.
  - Can be still in-order completion to avoid issues with precise exceptions, etc

VLIW architectures

- Very long instruction word (VLIW) processing provides significant parallelism.
- Rely on compilers to identify parallelism.
- VLIW requires considerably more sophisticated compiler technology than traditional architectures---must be able to extract parallelism to keep the instruction pipelines full.
- VLIW is popular for various embedded designs
  - EPIC = Explicitly parallel instruction computing.
    - Used in Intel/HP Merced (IA-64) machine.

Difference between microcontrollers, microprocessors and FPGA systems

- FPGA systems often contain CPUs in softcore (synthesized) or hardcore (part of die) format but can also contain logic blocks for other hardware, e.g., state machines, etc
- Microcontrollers are more limited in functionality and often do not include support for virtual memory and caches
  - Up to 50MHz
- Microprocessors are more performance capable and have typically virtual memory support
  - From 50MHz to GHz

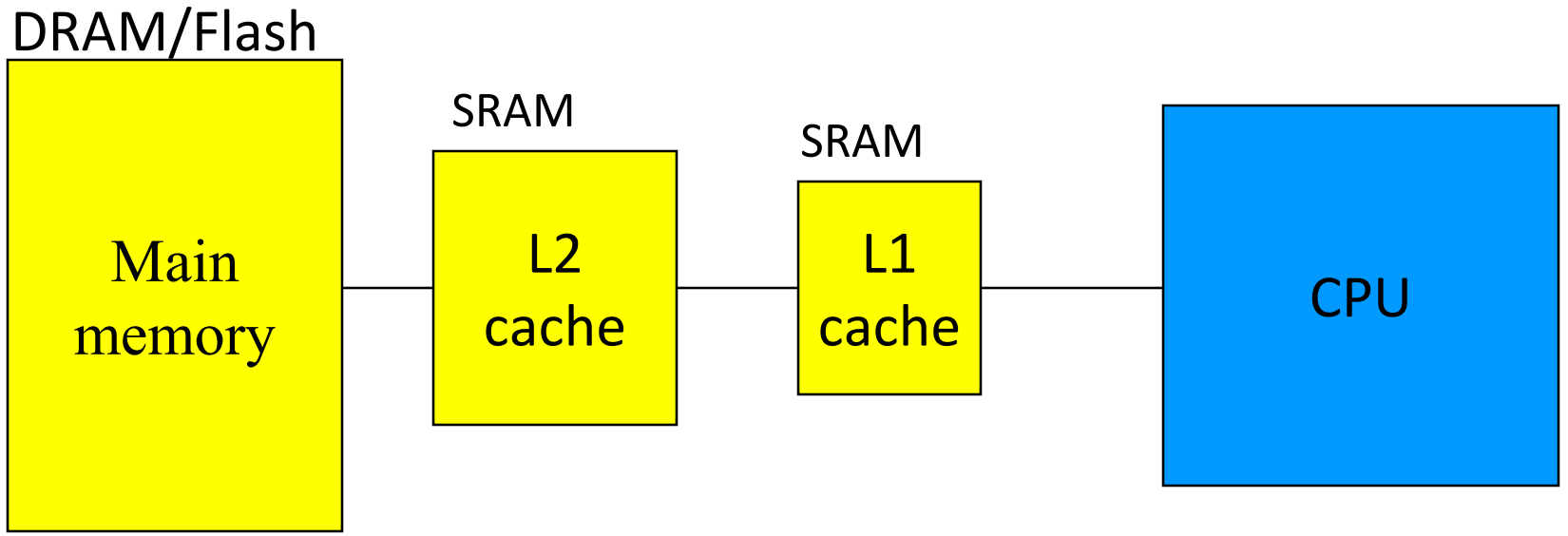


Another perspective: PLDs, FPGAs, ASICs, Structured ASICs

- Programmable logic devices (PLDs) provide low/medium density logic.
- Field-programmable gate arrays (FPGAs) provide more logic and multi-level logic.
- Application-specific integrated circuits (ASICs) are manufactured for a single purpose.
- Structured ASICs (see of gates wired together) are in between FPGAs and ASIC – manufactured for single purpose but manufacturing cheaper than ASIC since customization is often through metal layers (less mask costs)

Memory system

- CPU fetches data, instructions from a memory hierarchy:



- Some systems also include TLB/MMU to provide a cache during address translation and access control checks

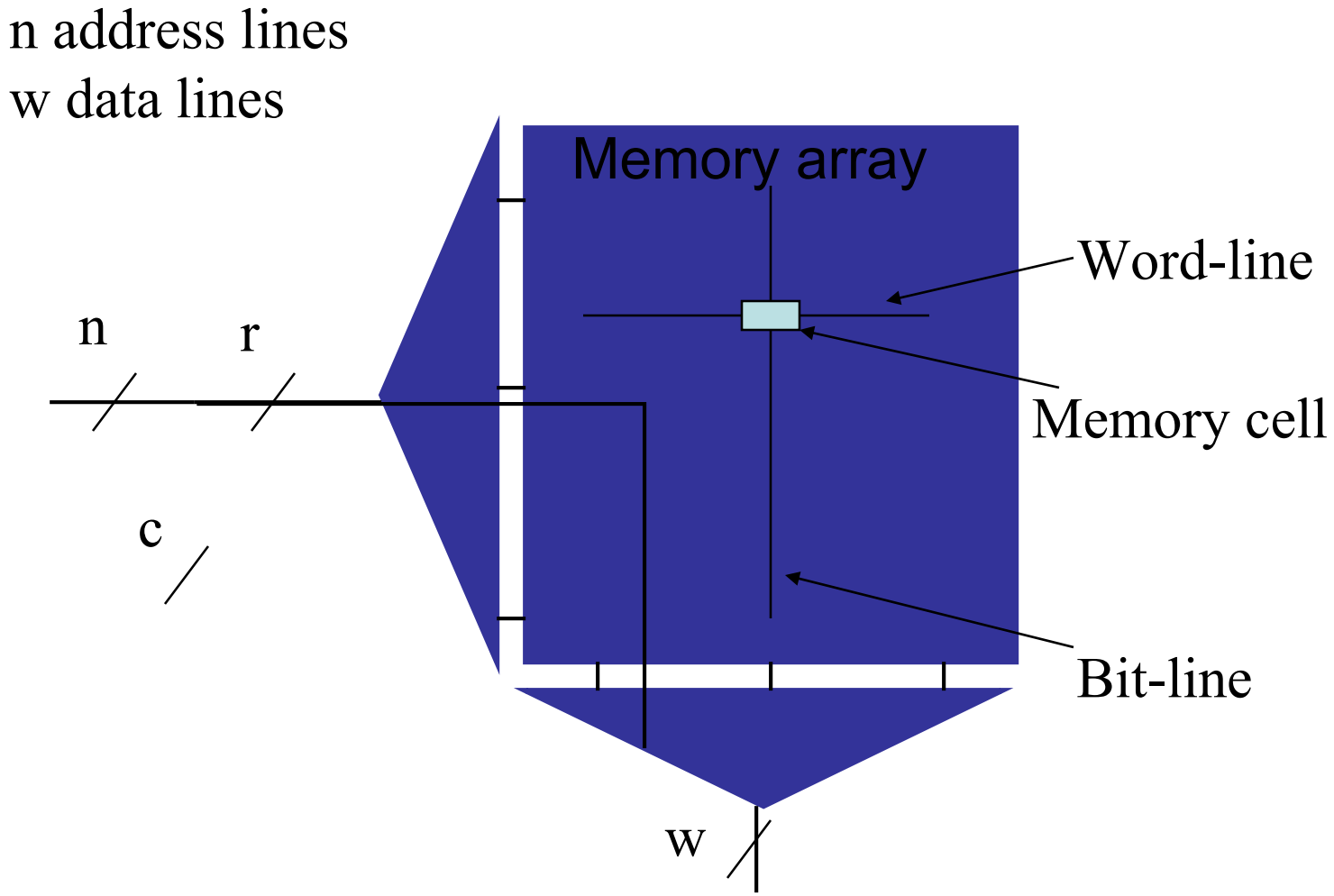
Memory hierarchy complications

- Program behavior is much more state-dependent.
  - E.g., depends on how earlier execution left the cache.
- Execution time is less predictable.
  - Memory access times can vary by 100X.
    - L1 1-4c, L2 8-12c, DRAM ~100c
  - Virtual memory, TLB, etc all affect performance
    - TLB hit vs miss, page hit vs miss

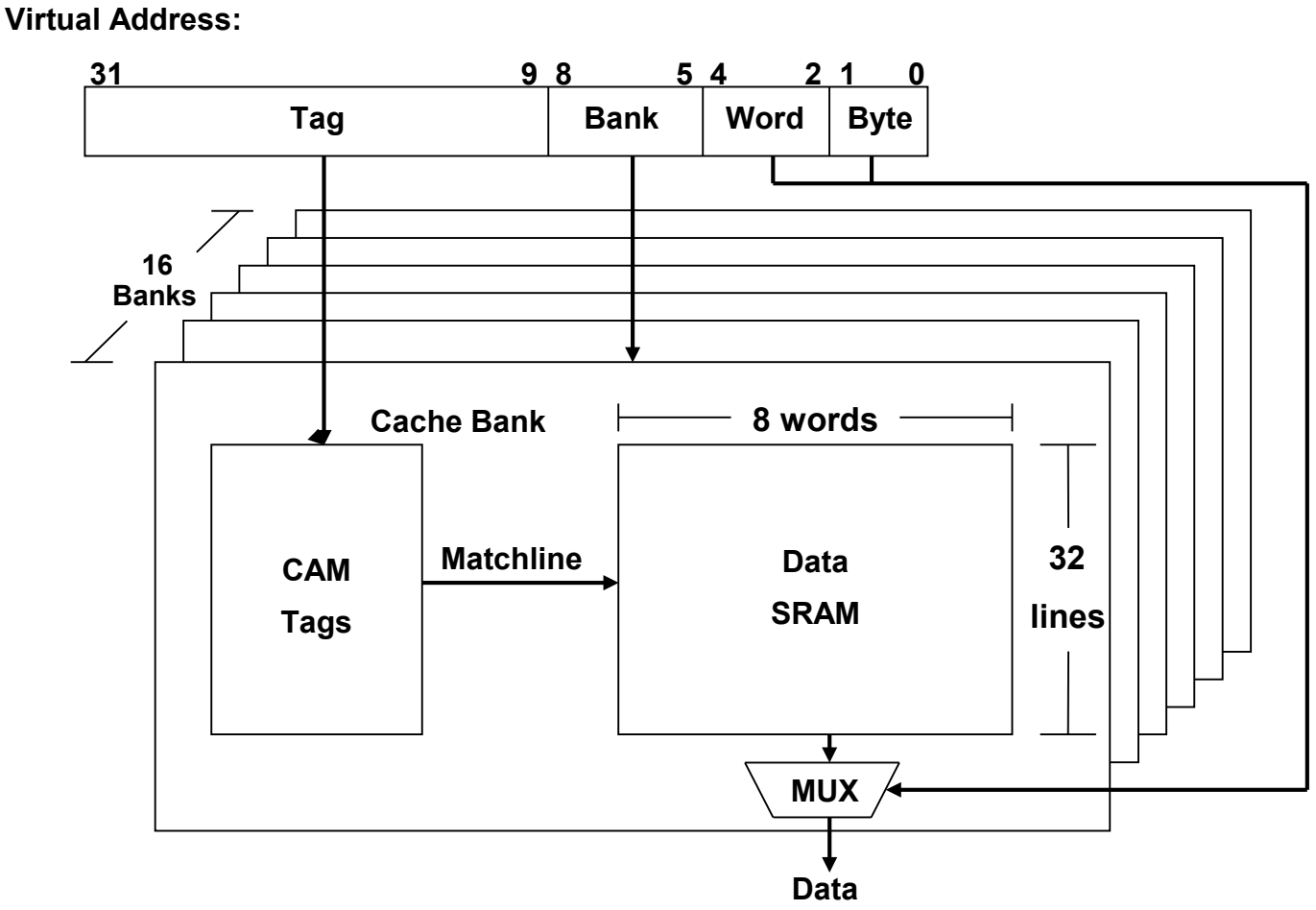
Types of memory

- ROM:
  - Mask-programmable.
    - High density – downside?
  - Flash programmable.
    - Challenges? Advantages? # of erase cycles
- RAM:
  - DRAM. Access time?
  - SRAM. Access time?

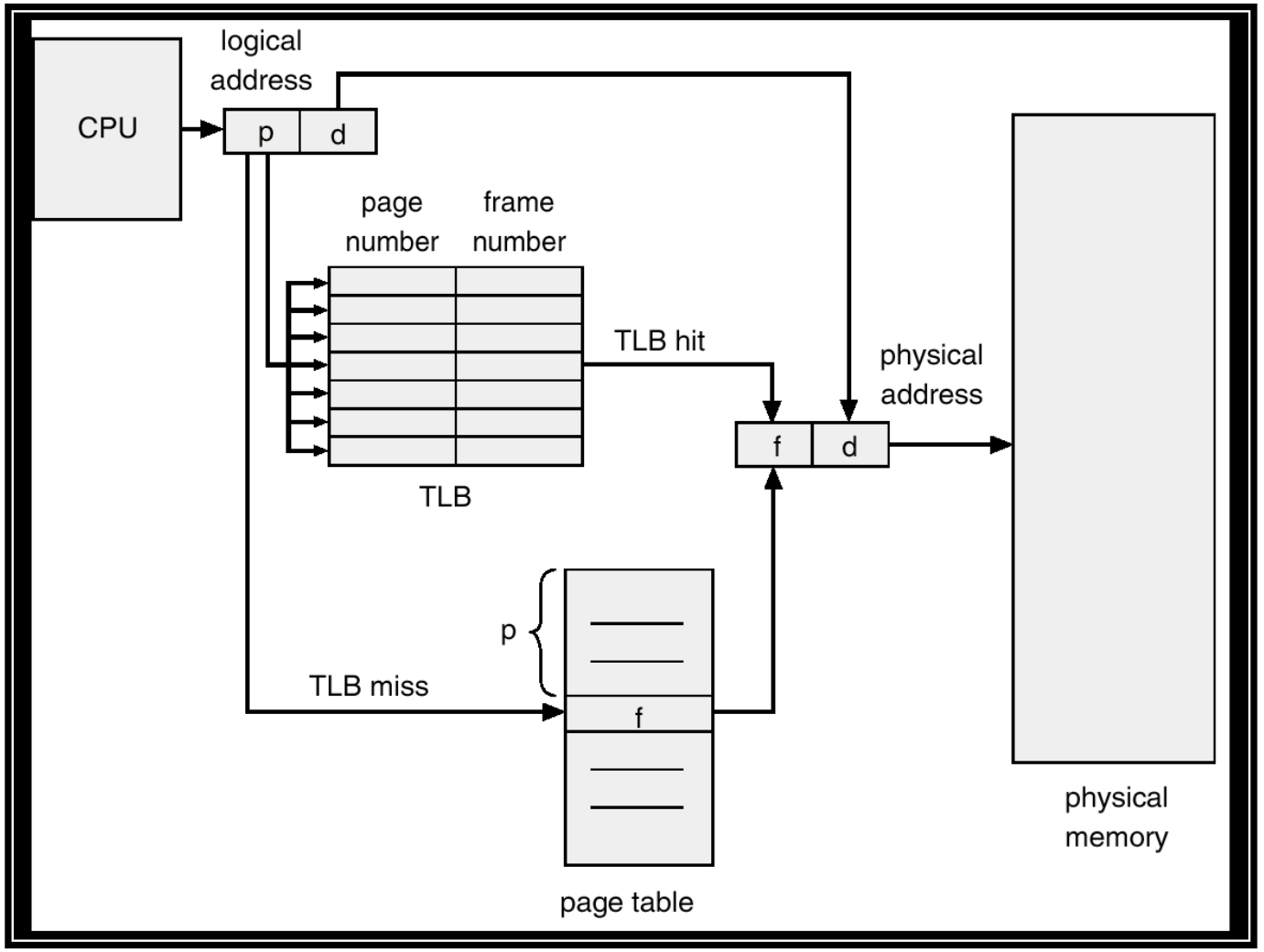
Memory device organization (e.g., SRAM block)



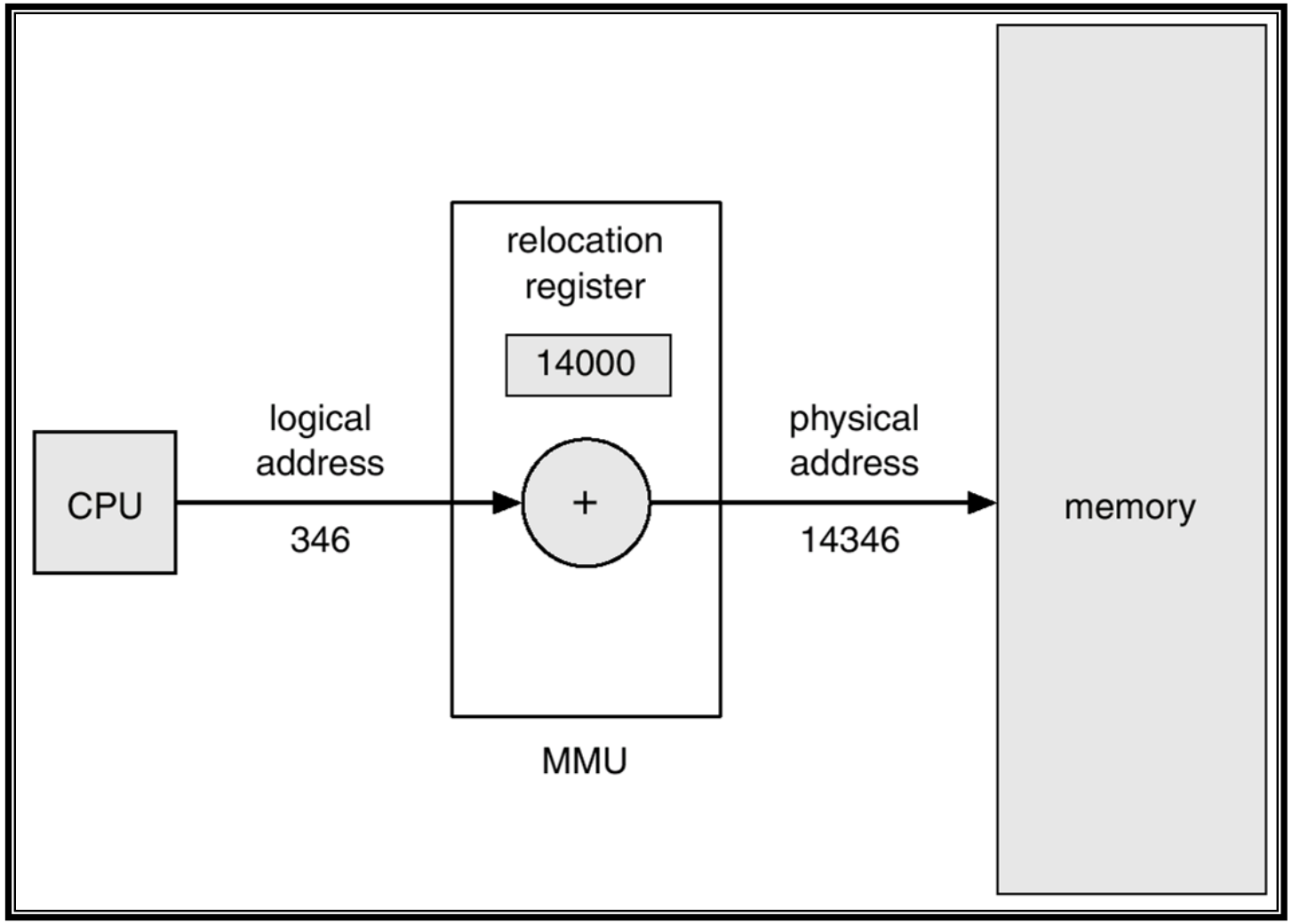
Cache Organization



Virtual Memory Organization Example



In Embedded Designs MMU Can Look Much Simpler



But Also Fairly Complicated

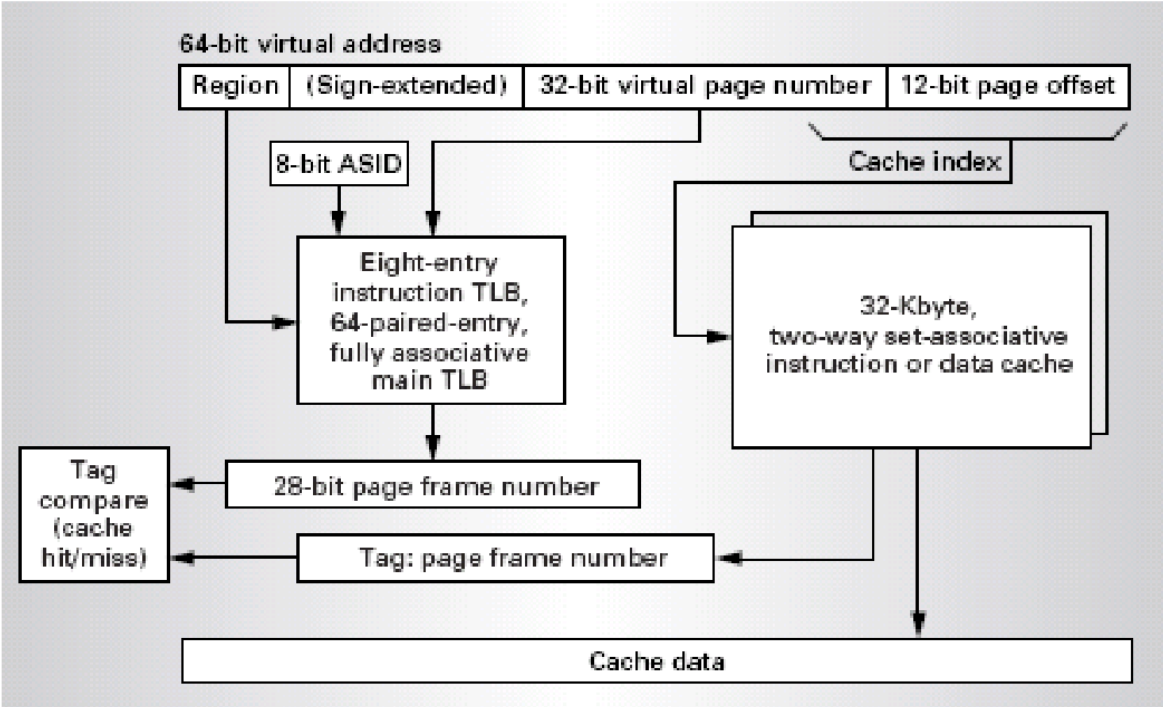


Figure 1. MIPS R10000 address translation mechanism. The split instruction and data caches are virtually indexed, both requiring an index larger than the page offset. The TLB lookup proceeds in parallel with the cache lookup. The earliest MIPS designs had physically indexed caches, and if the cache was larger than the page size, the cache was accessed in series with the TLB. ASID: address space identifier.

Programming model in Processors

- Assembly language
  - One-to-one with machine instructions (more or less).
  - Labels provide names for addresses (usually in first column).
  - Pseudo-ops: constants, define storage, define address

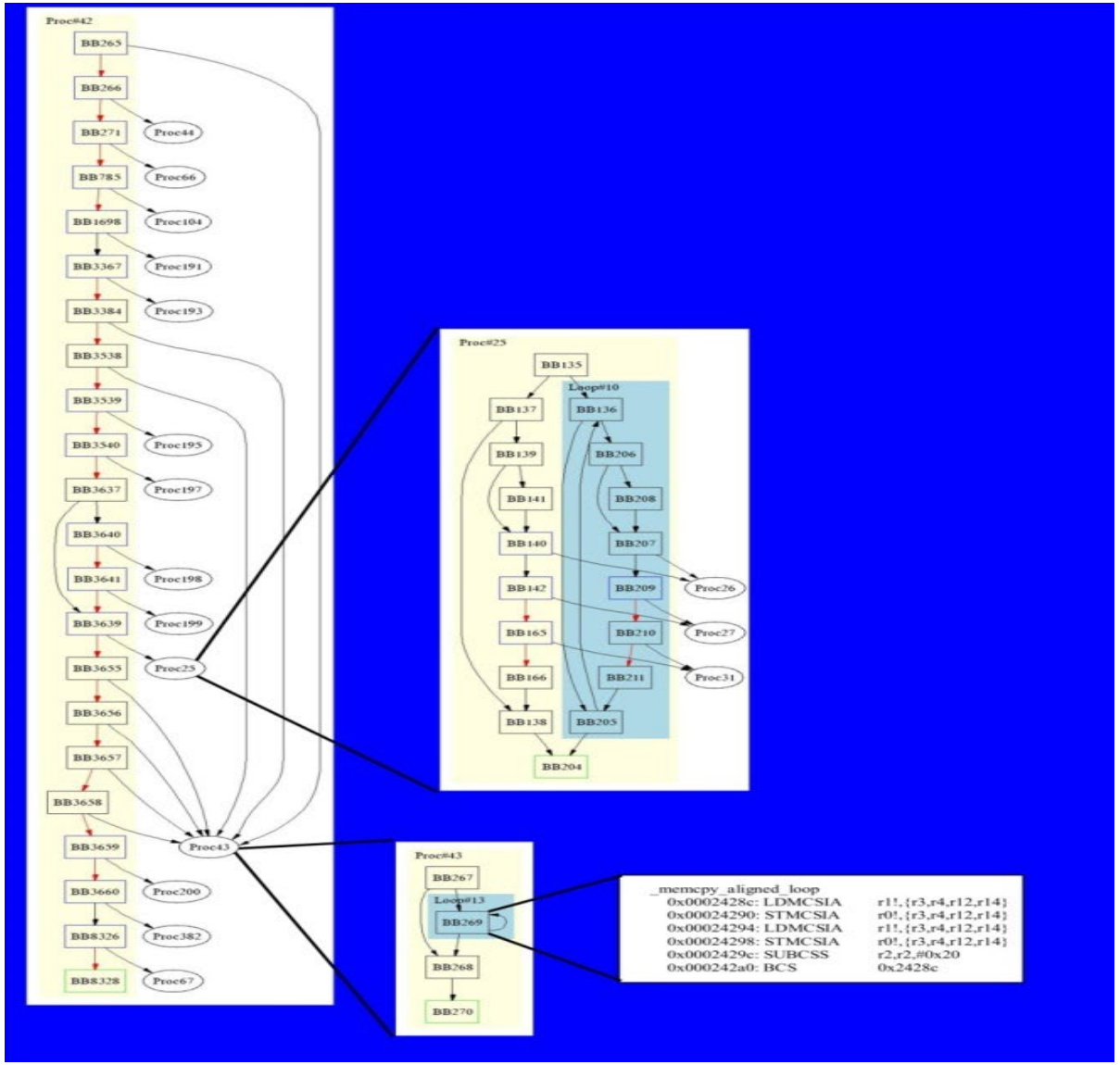
- Programming model: registers visible to the programmer.
  - For example ARM has 32 registers
  - Some registers are not visibible: system registers

ARM assembly language example

```
label1  ADR r4,c
        LDR r0,[r4] ; a comment
        ADR r4,d
        LDR r1,[r4]
        SUB r0,r0,r1 ; comment
```

Visualizing

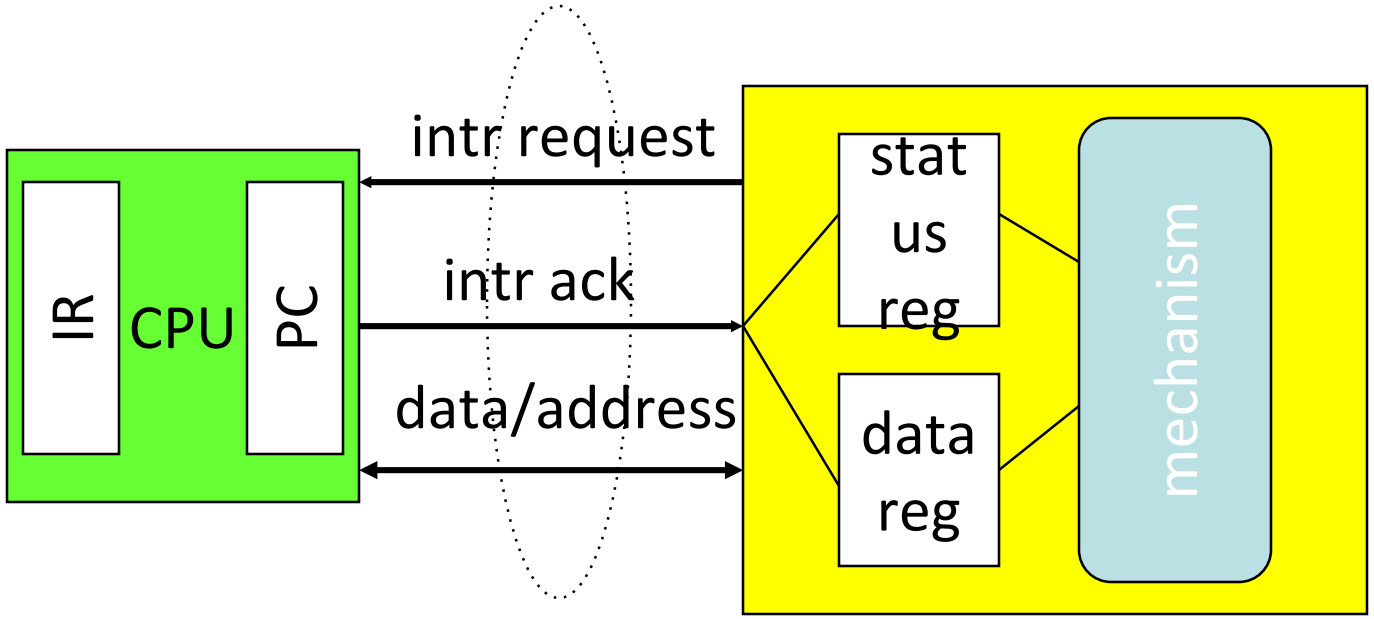
- Control Flow Graphs
- Procedures
- Loops
- Basic Blocks
- Instructions



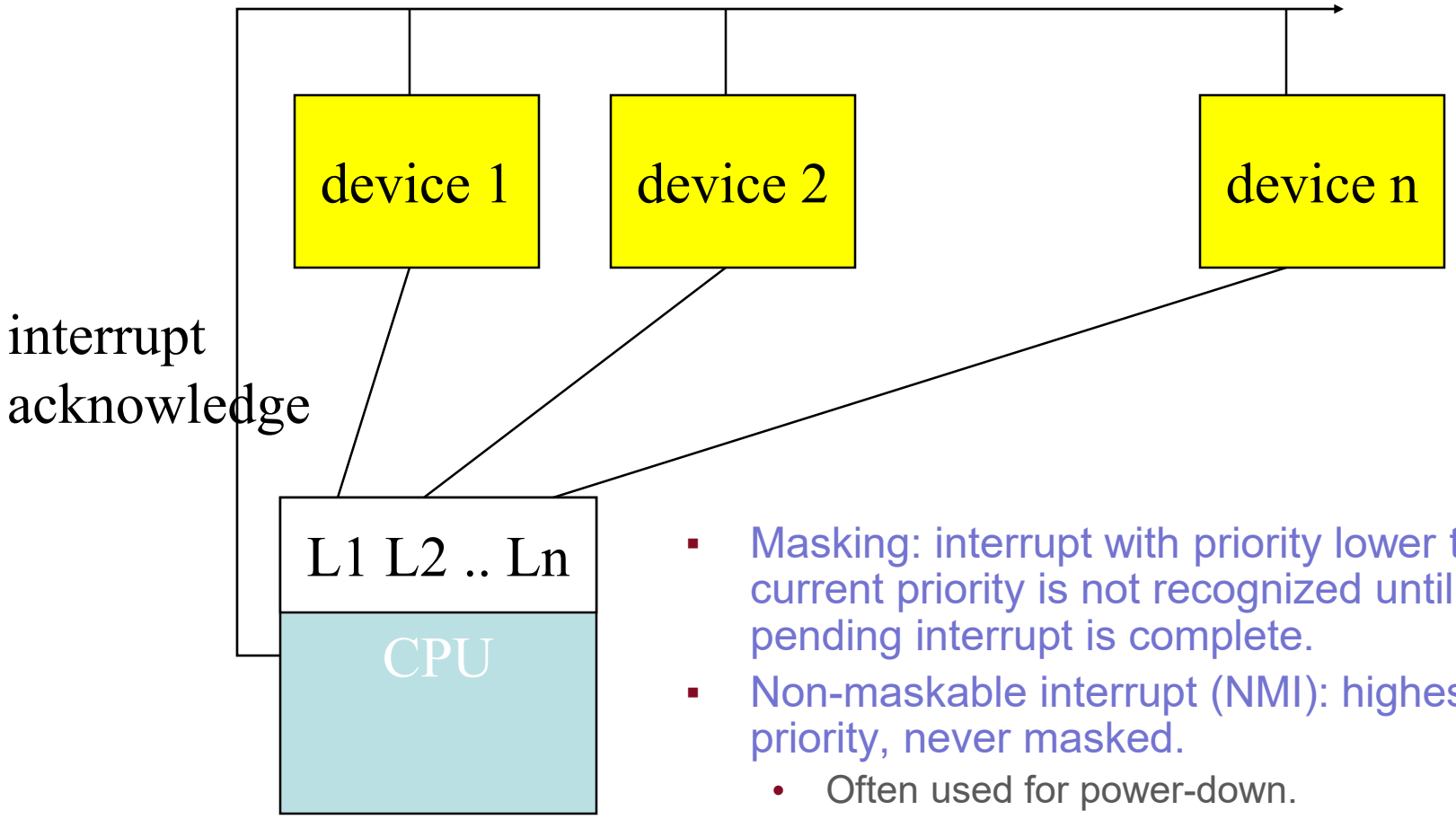
Outline for Today's Lecture

- Administrative, Lab2, Feedback on Lab1
- Embedded CPUs
- Interrupts
- IO
- Embedded System Software Design
- Basics on Image Manipulation

Interrupt interface

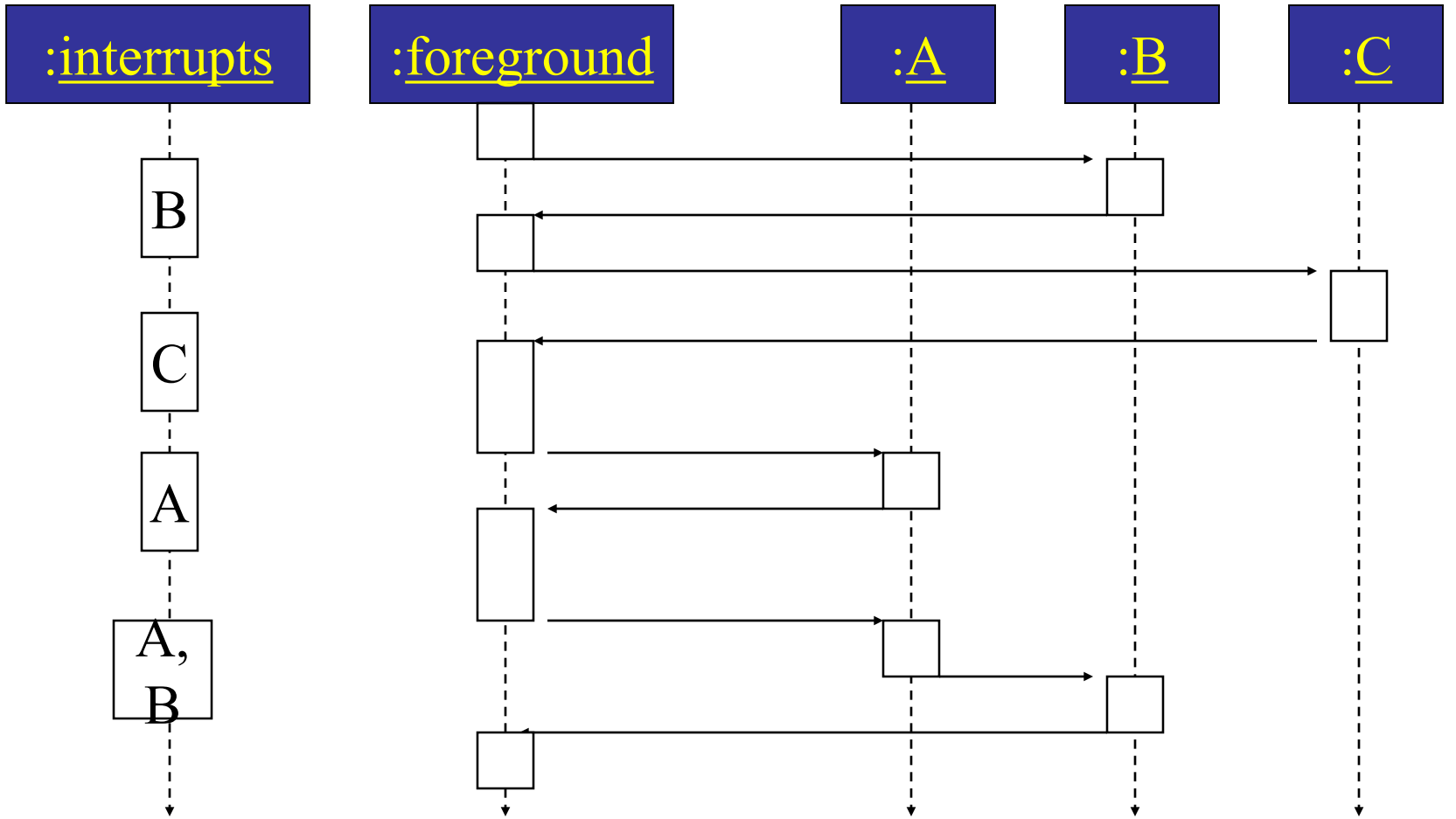


Prioritized interrupts

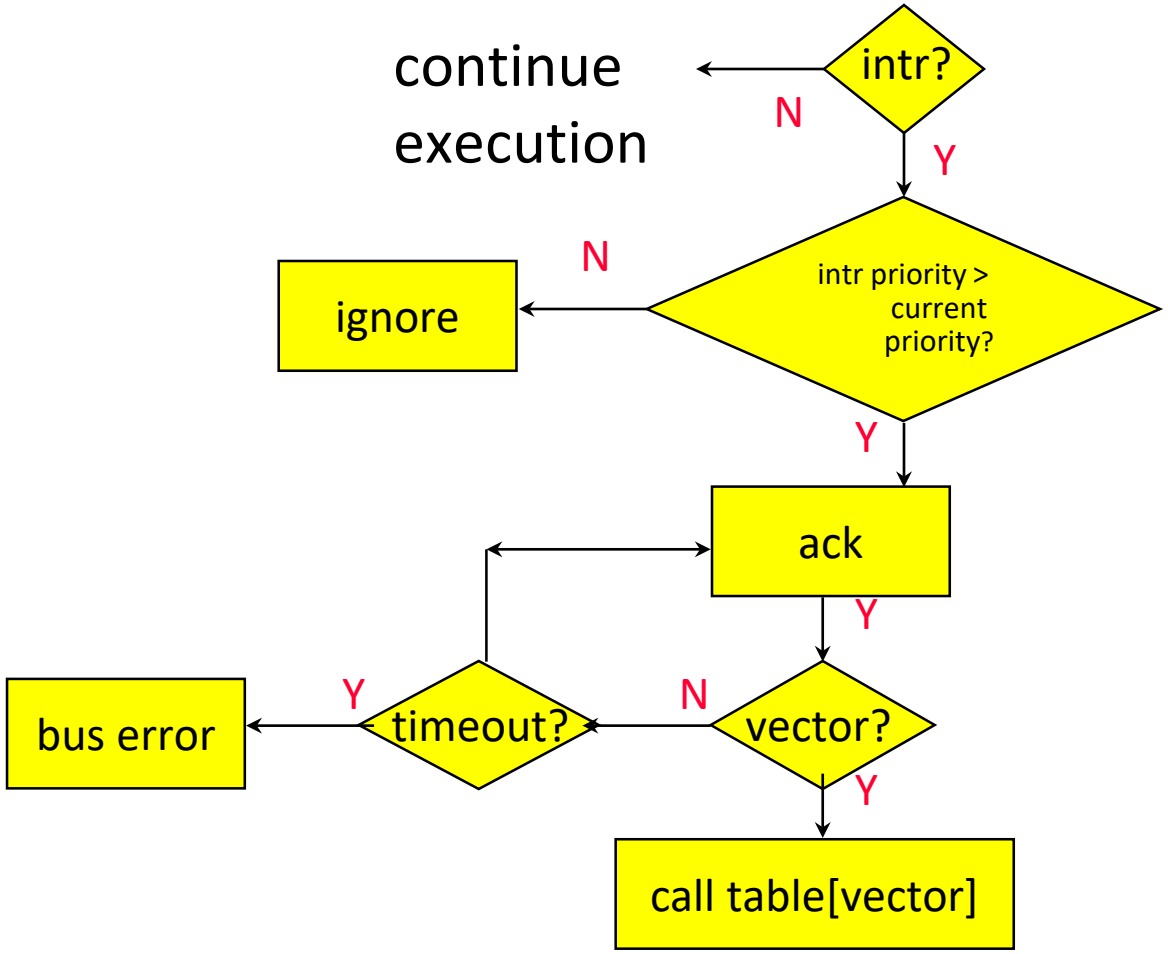


- Masking: interrupt with priority lower than current priority is not recognized until pending interrupt is complete.
- Non-maskable interrupt (NMI): highest-priority, never masked.
  - Often used for power-down.

Example: Prioritized I/O

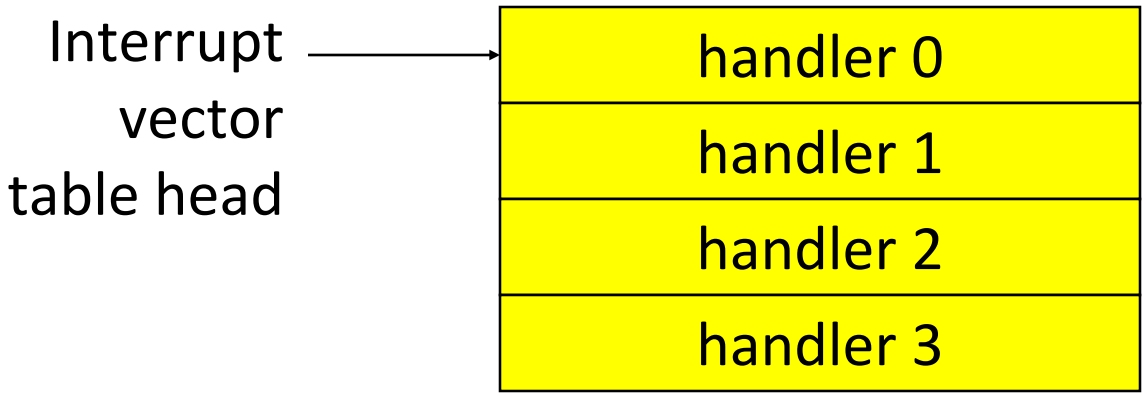


Generic interrupt mechanism



Interrupt vectors

- Allow different devices to be handled by different code.
- Interrupt vector table:



Sources of interrupt overhead

- Handler execution time.
- Interrupt mechanism overhead.
- Register save/restore.
- Pipeline-related penalties.
- Cache-related penalties.

ARM interrupts

- ARM7 supports two types of interrupts:
  - Fast interrupt requests (FIQs).
  - Interrupt requests (IRQs).
- Interrupt table starts at location 0.

Exception and Trap

- Exception: internally detected error.
  - Exceptions are synchronous with instructions but unpredictable.
  - Build exception mechanism on top of interrupt mechanism.
- Trap (software interrupt): an exception generated by an instruction.
  - Call supervisor mode.
  - ARM uses SWI instruction for traps.

Outline for Today's Lecture

- Administrative, Lab2, Feedback on Lab1
- Embedded CPUs
- Interrupts
- IO
- Embedded System Software Design
- Basics on Image Manipulation

I/O devices

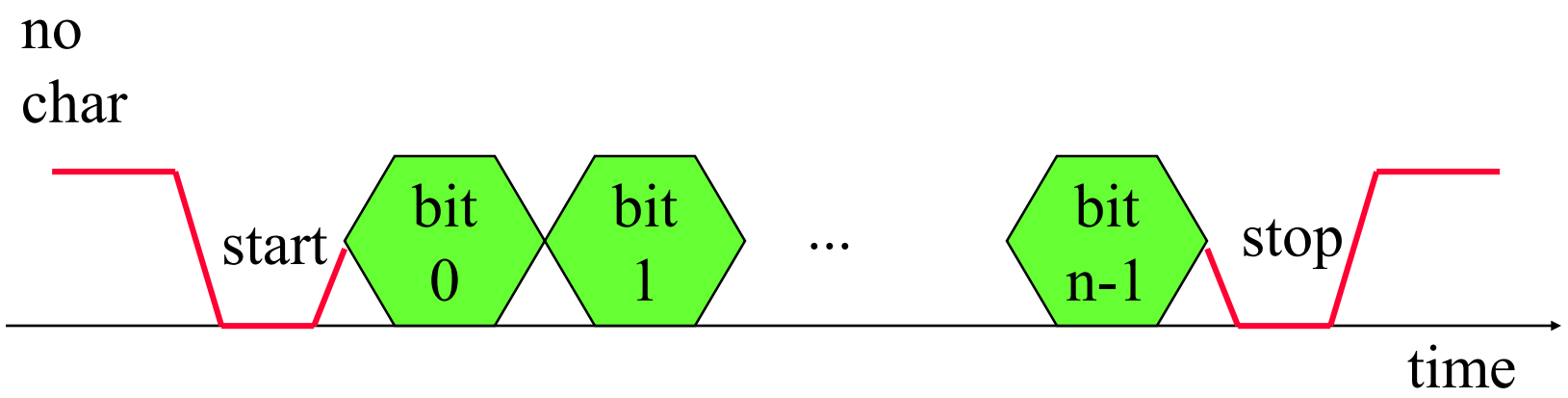
- I/O devices:
  - serial links (UART, USB, SPI, RS-232, RS-485, I2C, S2C, high-speed such as PCIe…)
  - timers and counters
  - keyboards
  - displays
  - analog I/O

Application: 8251 UART

- Universal asynchronous receiver transmitter (UART) : provides serial communication.
- 8251 functions are integrated into standard PC interface chip.
  - 8255 used to be a parallel interface
- Allows many communication parameters to be programmed.

Serial communication

- Characters are transmitted separately:



Serial communication parameters

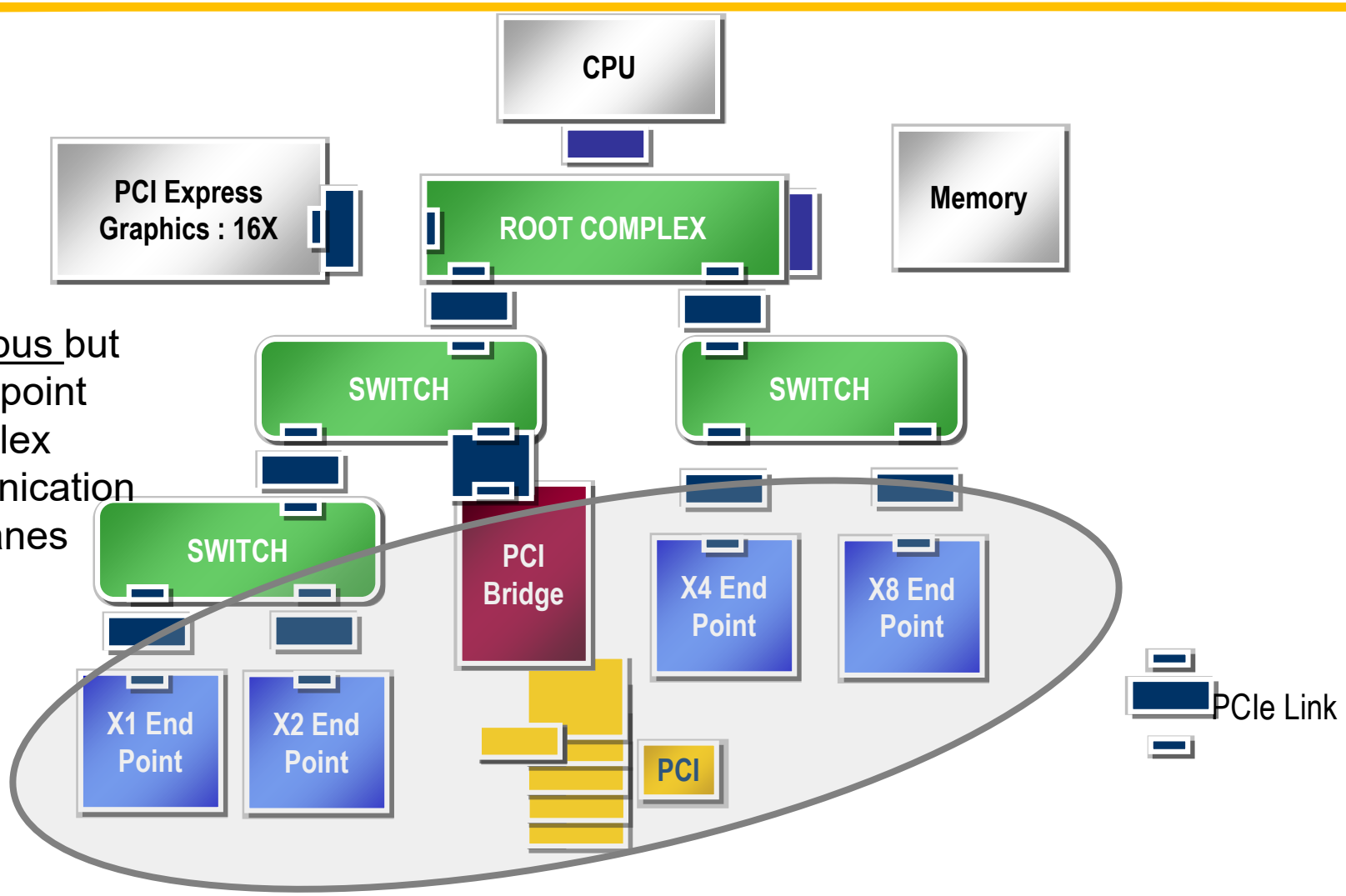
- Baud (bit) rate.
- Number of bits per character.
- Parity/no parity.
- Length of stop bit (1, 1.5, 2 bits).

Programming I/O

- Two types of instructions can support I/O:
  - special-purpose I/O instructions;
  - memory-mapped load/store instructions.
- Intel x86 provides in, out instructions. Most other CPUs in the embedded space use memory-mapped I/O.
- I/O instructions do not preclude memory-mapped I/O.

PCIe Architecture

- Not a bus but point to point Full duplex Communication
- 1-32 lanes each 32MB/s



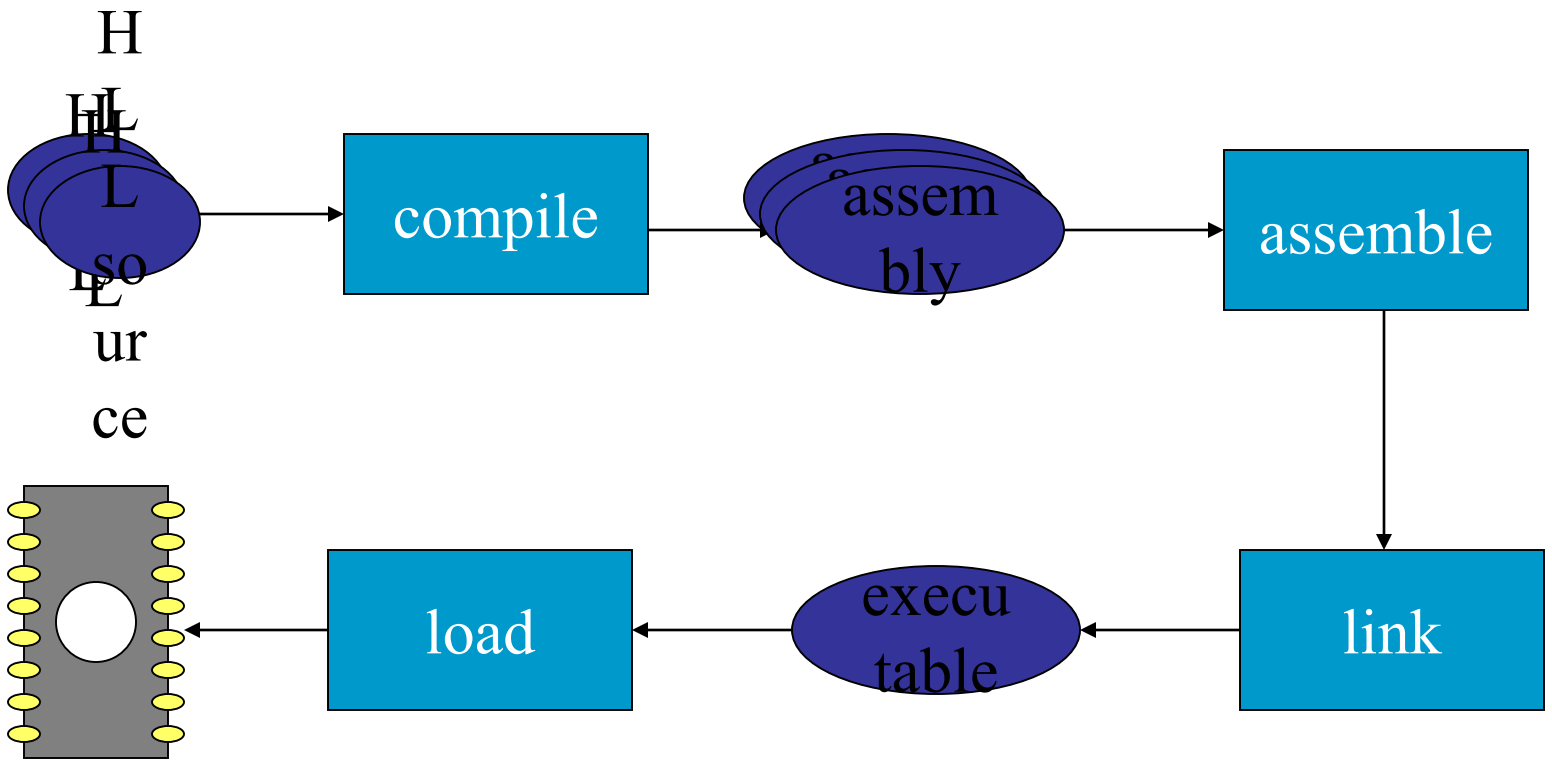
Outline for Today's Lecture

- Administrative, Lab2, Feedback on Lab1
- Embedded CPUs
- Interrupts
- IO
- Embedded System Software Design
- Basics on Image Manipulation

Software components

- Need to break the design up into pieces to be able to write the code.
- Some component designs come up often.
  - A design pattern is a generic description of a component that can be customized and used in different circumstances.
- Firmware – basic system software for a chip
- Device driver – control of hardware modules in a system tightly coupled with operating systems

Building Code Image



Assemblers

- Major tasks:
  - Generate machine code for symbolic instructions;
  - translate labels into addresses;
  - handle pseudo-ops.

Linking

- Combines several object modules into a single executable module.

Dynamic linking

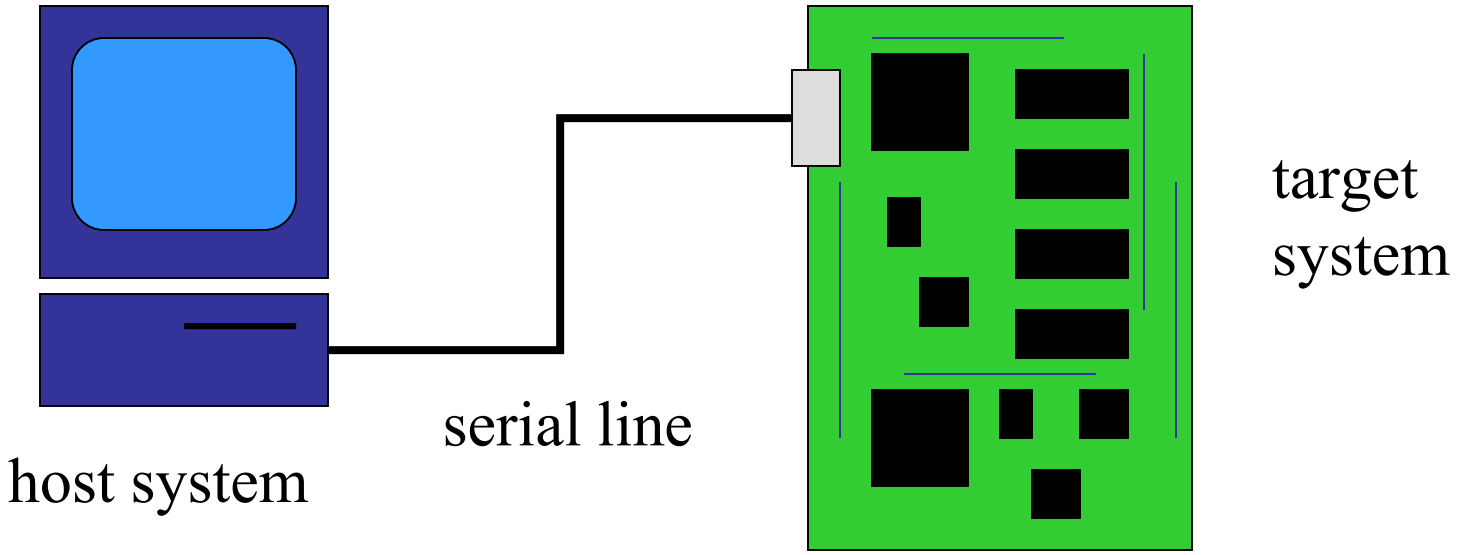
- Some operating systems link modules dynamically at run time:
  - shares one copy of library among all executing programs;
  - allows programs to be updated with new versions of libraries.

What's Next ?

- Lab 2 details by TA.

How to build/debug? Host -target design, debugging

- Use a host system to prepare software for target
  - Cross compiler: compiles code on host for target system.
- Cross debugger: displays target state, allows target to be controlled.



In-circuit emulators

- A microprocessor in-circuit emulator is a specially-instrumented microprocessor.
  - Allows you to stop execution, examine CPU state, modify registers.

- Many microprocessors also support software debugging through special ports and (typically) Jtag interface

Outline for Today's Lecture

- Administrative, Lab2, Feedback on Lab1
- Embedded CPUs
- Interrupts
- IO
- Embedded System Software Design
- Basics on Image Manipulation

Images

- A picture is a rectangular array of pixels (=picture element)
- A digital picture is a picture stored in binary (bits).
  - The picture resides in a digital storage system as a file.

Pixels, Grey Values and Quantization

- Conceptually, a monochrome (black and white) image is a function f(x, y), sampled over a two-dimensional grid.
  - Each sample value is called a pixel (picture element).

- Conceptually, the function is real-valued and has a continuous range. This is called the grey value of the pixel.
  - On a computer, it is represented with a finite number of bits. This is called quantization.

Example

Raw picture format, 256x256, 1 byte per pixel

Pixel sequence in file

1	2	3	…	256
257	258	259	…	512
513	514	515	…	768
…	…	…	…	…
65,280	65,281	…	…	65,536

Calculations

- In the so-called raw format, the file contains only the gray values of the pixels.
- Bits/picture = Rows x Columns x bits/pixel
- Bytes/picture = Rows x Columns x bytes/pixel
- Example:
  - For the previous slide, 256 rows, 256 columns, 1 byte per pixel.
  - Bytes = 256x256x1 = 65536

Pixels, Quantization, and Quality

- A given picture can be represented with different numbers of pixels and various numbers of bits per pixel.
  - Fewer pixels produces lower quality
  - Fewer bits per pixel produces lower quality

- There is a tradeoff between quality and picture storage requirements.

Examples of quantization vs. resolution

256x256, 8 bit, 64 kB

256x256, 4 bit, 32 kB

256x256, 2 bit, 16 kB

256x256, 1 bit, 8 kB

64x64, 8 bit, 4 kB
Lower resolution

Some Image Processing functions

- Add timestamp onto image
- Counter to keep track of number of pictures taken
- Rotate, mirror, invert image
- Simple edge detection (hard)
- Detect motion in images (harder)
- Detect illegal behavior in images (computer vision)

Uses of Digital Pictures

- Consumer and Commercial Photography
- Web-cam
- Medical Imaging
  - X-ray
  - Tomography
  - Ultrasound
- Video
  - DVD
  - Broadcast
  - Surveillance

What quality levels are required for each?
Storage requirements?

Why Standard Formats?

- Interoperability
  - Image made by Nikon, viewed on computer made by Apple.
- Advantages of standards
  - Competition among vendors (lower prices)
  - Creation of markets
  - Multiple vendors - product cycle safety

Image Coding (Compression)

- Why compress?
  - Store more pictures in same memory
  - Spend less time sending picture over web
- Lossless compression (LZ, Huffman, RLE):
  - Decompress file and get the same picture, bit - for - bit
- Lossy compression (JPEG,…):
  - Decompress and get something similar.
  - Any amount of compression is possible.
  - Tradeoff between image quality and compression.

Example of JPEG compression

Very high quality:
compression =
2.33
Photoshop Image

Very low quality:
compression = 115
Produced by
MATLAB

USB

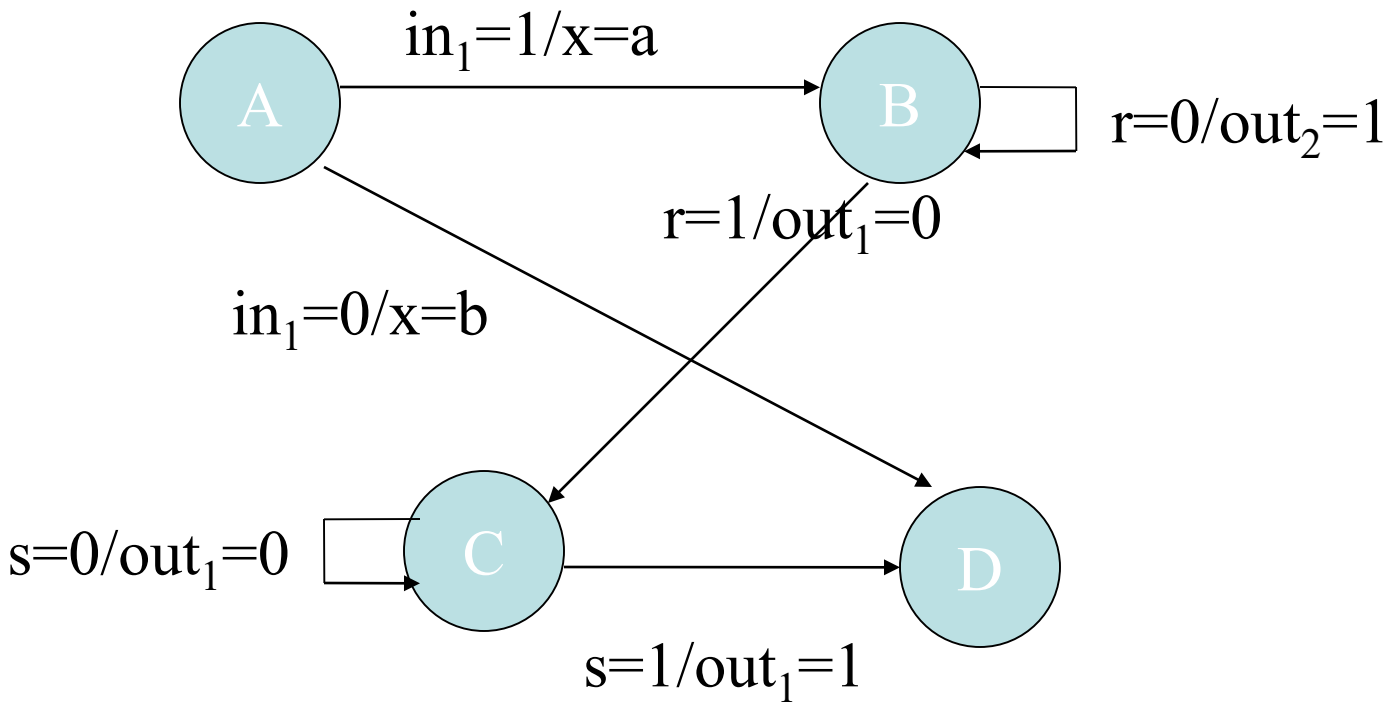
- Goals:
  - Easy to use. USB-C latest.
  - Low cost for consumer devices.
  - Up to 4.8Gb/s in USB 3.0; 10x better than 2.0; USB-C up to 10Gb/s (v 3.1 protocol, 2x speed vs 3.0) & up to 100W power, and reversible, plus support for Displayport, Thunderbolt 3 (40Gb/s)
  - History
    - A Low Speed (1.1, 2.0) rate of 1.5 Mbit/s(187 KB/s) that is mostly used for Human Interface Devices (HID) such as keyboards, mice, and joysticks.
    - A Full Speed (1.1, 2.0) rate of 12 Mbit/s (1.5 MB/s). Full Speed devices divide the USB bandwidth between them in a first-come first-served basis and it is not uncommon to run out of bandwidth with several isochronous devices. All USB Hubs support Full Speed.
    - A Hi-Speed (2.0) rate of 480 Mbit/s (60 MB/s).
    - A Super-Speed (3.0) rate of 4.8 Gbit/s (600 MB/s).
    - USB 3.1 10Gb/s, USB 3.2 20Gb/s

USB bus protocol

- Polled bus, all transfers initiated by host.
  - USB On-the-go (UTG) allows USB devices to act as a host
- Basic transaction:
  - Host sends token packet:
    - Type and direction.
    - USB device number.
    - Endpoint number (subdevice).
  - Data transfer packet.
  - Acknowledge packet.

Embedded system design often includes software state machines

- State machine keeps internal state as a variable, changes state based on inputs.



C state table

```
switch (state) {
case A: if (in1==1) { x = a; state = B; }
  else { x = b; state = D; }
   break;
case B: if (r==0) { out2 = 1; state = B; }
  else { out1 = 0; state = C; }
   break;
case C: if (s==0) { out1 = 0; state = C; }
  else { out1 = 1; state = D; }
   break;
```