PROCESADORES DIGITALES

Analog versus Digital Control

Analog control

Advantages

- High bandwidth.
- High resolution.
- No data conversions required.
- Analysis and design methods are well known.
- Adjustment by potentiometers or variable capacitors are easy and fast.

Disadvantages

- Temperature drift: Control performance depends on passive and active components characteristics that change with temperature.
- Component aging: periodic adjustments are required to maintain good performance.
- Hardware design: modifications, adaptations or upgrades mast be done at hardware level.
- Can implement only simple designs (PID, lead-lag).
- Sensitive to noise.
- No communication capability.
- No effective storage capability.

Digital control

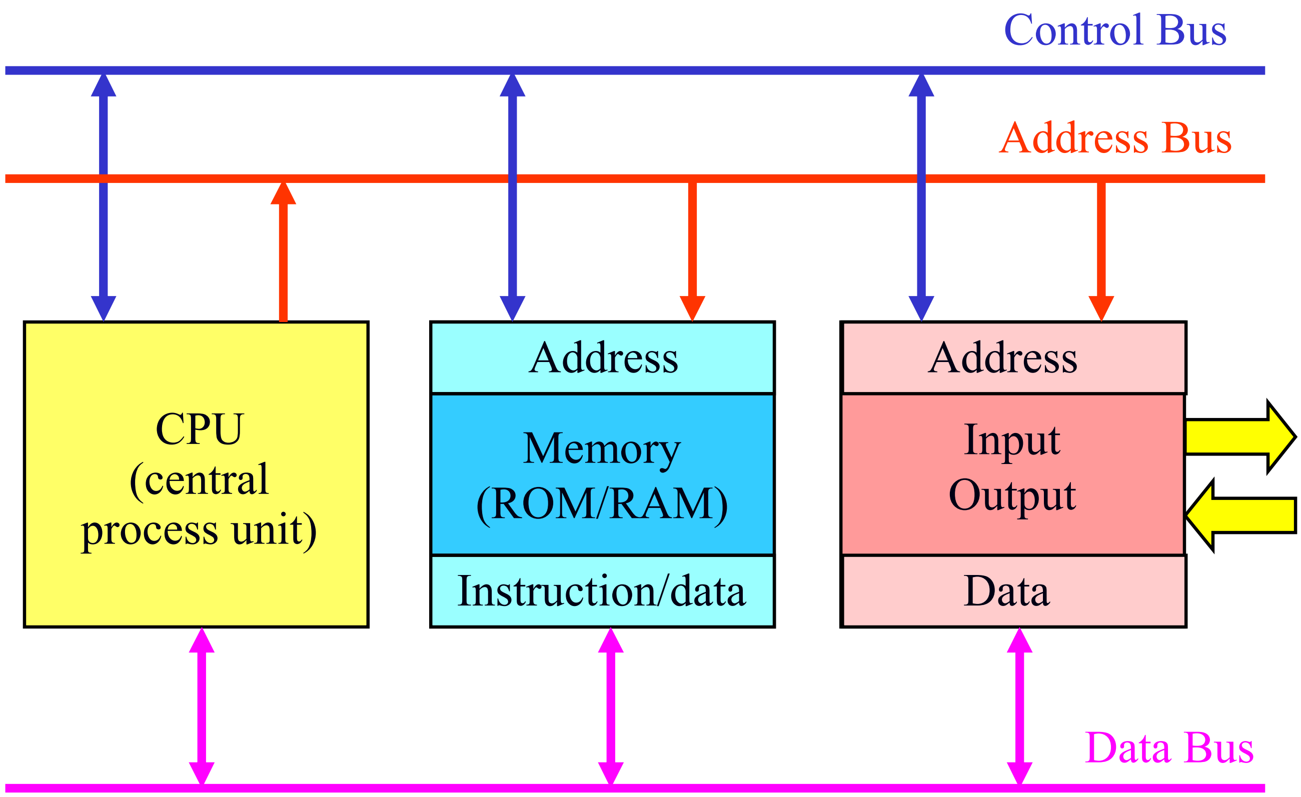
Advantages

- Programmable solution: modifications, adaptations and upgrades are done by software.
- Less sensitive to the environment.
- Can implement advanced control algorithms.
- Capable of self-tuning, adaptive control, and nonlinear control functions.
- Capable of sensorless-operation (they are replaced by estimators or observers).
- Capable of additional functions as monitoring, diagnosis, protections, etc.
- Communication capability: can be incorporated into a network control system.
- Flexible storage capability.

Disadvantages

- Data conversion are required.
- Analysis and design methods are more complex.
- Sampling and resolution can affect the disturbance rejection capability.
- Computation delay limits the system bandwidth and can affect stability.
- Quantization errors can reduce the control precision.

Microprocessor Architecture: Basic Architecture (Von Newmann)



CPU: controls de computer operation and executes the instructions.

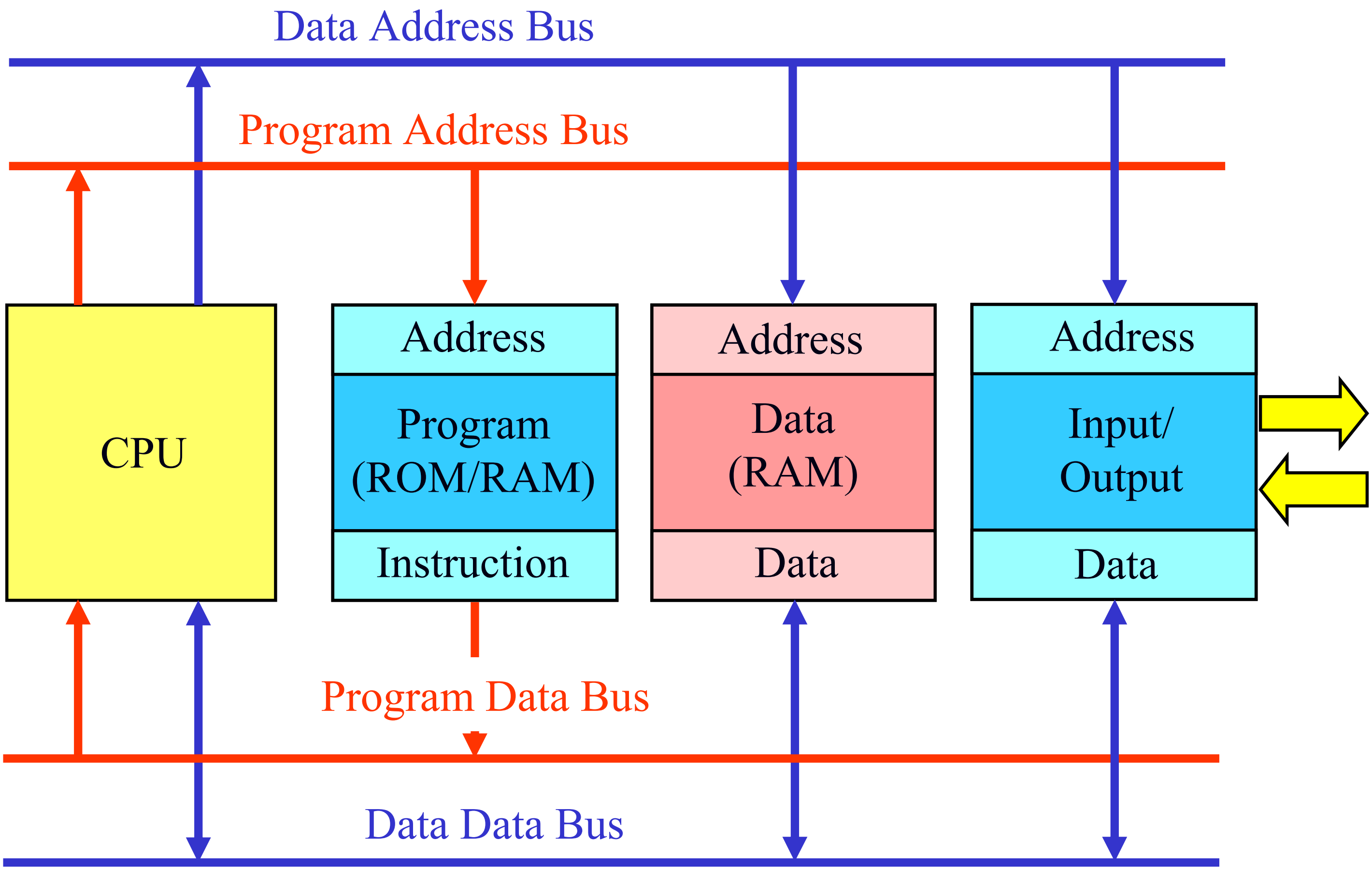
Memory: Contains instructions and data (necessary for the program or produced by the program).

Input/Output Unit: Interfaces the CPU with the outside world (External memory, A/D and D/A converters, serial or parallel communications, digital inputs/outputs, etc.

Limitations:

- Instructions and data travel over the same bus.
- The microcomputer throughput is limited by the bus bandwidth and the memory access time.

Microprocessor Architecture: Harvard Architecture



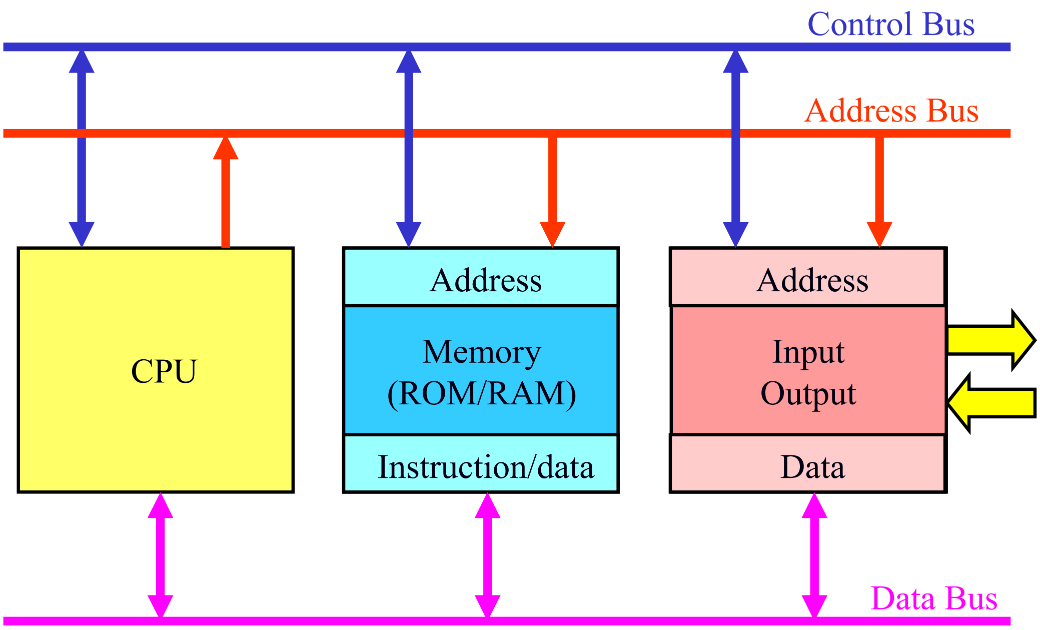
CPU: controls de computer operation and executes the instructions.

Memory: Contains instructions and data (necessary for the program or produced by the program)

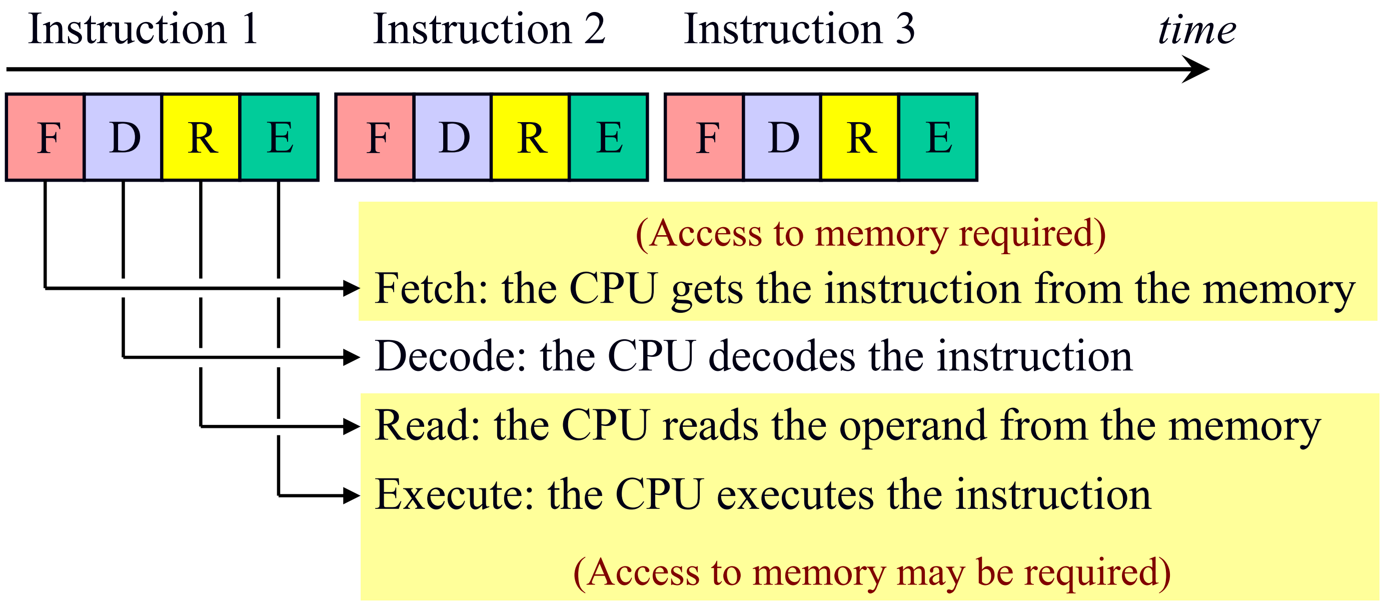
Input/Output Unit: Interfaces the CPU with the outside world (External memory, A/D and D/A converters, serial or parallel communications, digital inputs/outputs, etc.

Often separate buses are used to access memory and inputs/outputs (peripherals and external hardware)

Microprocessor Architecture: Basic Architecture



Instruction Cycle



(Access to memory required)

Fetch: the CPU gets the instruction from the memory

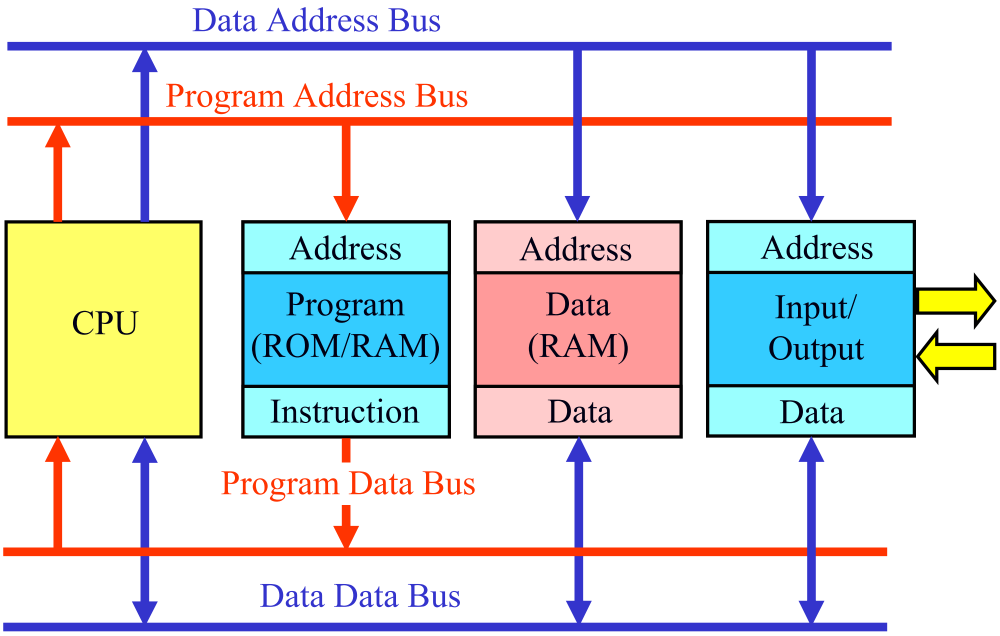
Decode: the CPU decodes the instruction

Read: the CPU reads the operand from the memory

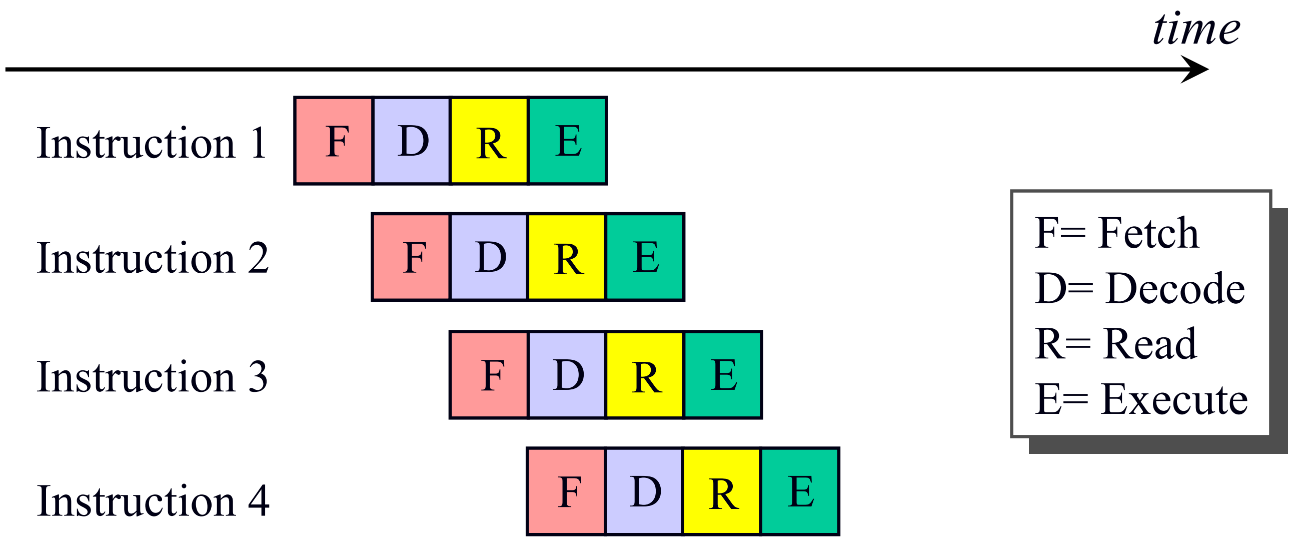
Execute: the CPU executes the instruction

(Access to memory may be required)

Pipelined Architectures



- The basic operations are assumed by separate units.
- Instructions overlap, so the CPU can handle several instructions at the same time.



F= Fetch
D= Decode
R= Read
E= Execute

- Multiple bus (Harvard) architecture is required.
- Most instructions require a single cycle.
- Instructions breaking the pipeline require four cycles, dramatically decreasing the efficiency. Its use should therefore be restricted as much as possible. Such instructions include:
    - ✓ Branches (conditional jumps)
    - ✓ Jumps
    - ✓ Calls to subroutines
    - ✓ Interrupts

Microprocessor Architecture: RAM Memory

RAM (Random-Access Memory)

- Read and write ☺.
- Volatile (data is lost when the power is shut off) ☹.

Static RAM (SRAM)

- Built with flip-flops.
- Density relatively low ☹.
- Short Access time (8 ns) ☺.

Dynamic RAM (DRAM)

- Data is stored in capacitors.
- Require periodic refresh to keep data valid ☹.
- High density ☺.
- Longer access time than that of static RAMs (20 to 100 ns) ☹.
- Commonly used as microcomputer main memory.

Microprocessor Architecture: ROM Memory

ROM (Read only memory)

- Mask programmed by the chip manufacturer.
- Typically used for storage of programs in a definitive version look-up tables, etc.

Programmable read-only memory (PROM)

- Contain fusible links connected with logic gates. Data are written blowing the links.
- Programmed by the user ☺.
- Programming is done off-board using a PROM programmer ☹.

Erasable programmable read only memory (EPROM)

- Can be reprogrammed a limited number of times ☺.
- Data is stored as electric charges is floating gate devices.
- The entire stored data can be erased exposing the ship to ultraviolet light.
- EPROMs must be removed from system and erased before reprogramming ☹.

Flash EPROM

- Is an EPROM with fast erasing mechanism.
- The entire stored data can be erased on-board using a short electric pulse ☺.
- Programming can be done on board ☺.

Electrically erasable programmable read only memory (EEPROM)

- Each byte can be erased and reprogrammed on-board by electric pulses ☺.

Microprocessors, Microcontrollers and DSP’s

Microprocessors

General purpose, off line data manipulation

- Reduced on-chip hardware
- Don’t include on-chip interrupt controller
- Not suitable for real-time control applications

DSP Controllers

Microcontrollers

Real time control

Von-Newman architecture (single bus)
8–16 bits bus, fixed point

Hardware incorporated (depends on model)

- Memory (RAM, ROM)
- A/D converters
- D/A converters
- General purpose PWM
- Three-phase PWM
- Timers
- Serial communications
- Interrupt controller
- High speed inputs
- High speed outputs
- DMA

DSP’s

Real time mathematical calculations

Harvard architecture (multiple bus)
16–32 bits, fixed and floating point

Hardware incorporated

- Memory (RAM, ROM)
- Timers
- Serial communications
- Interrupt controller
- DMA

Microprocessors, Microcontrollers and DSP's

RISC (Reduced Instruction Set Computing Processors)

- Reduced instruction set.
- Instructions are implemented directly in hardware.
- Reduced addressing modes.
- Pipelined architecture.
- Complex instructions implemented by software.
- Large register file.
- Instruction cache.
- Single-cycle execution.
- Simple format instructions.

CISC (Complex Instruction Set Computing Processors)

- Numerous instructions groups.
- Numerous addressing modes.
- Microcoded instructions (take several cycles to complete).

✓ Conventional Microprocessors and Microcontrollers are CISC.

✓ Some advanced processors show a RISC architecture.

✓ DSP's are closer to RISC architecture.

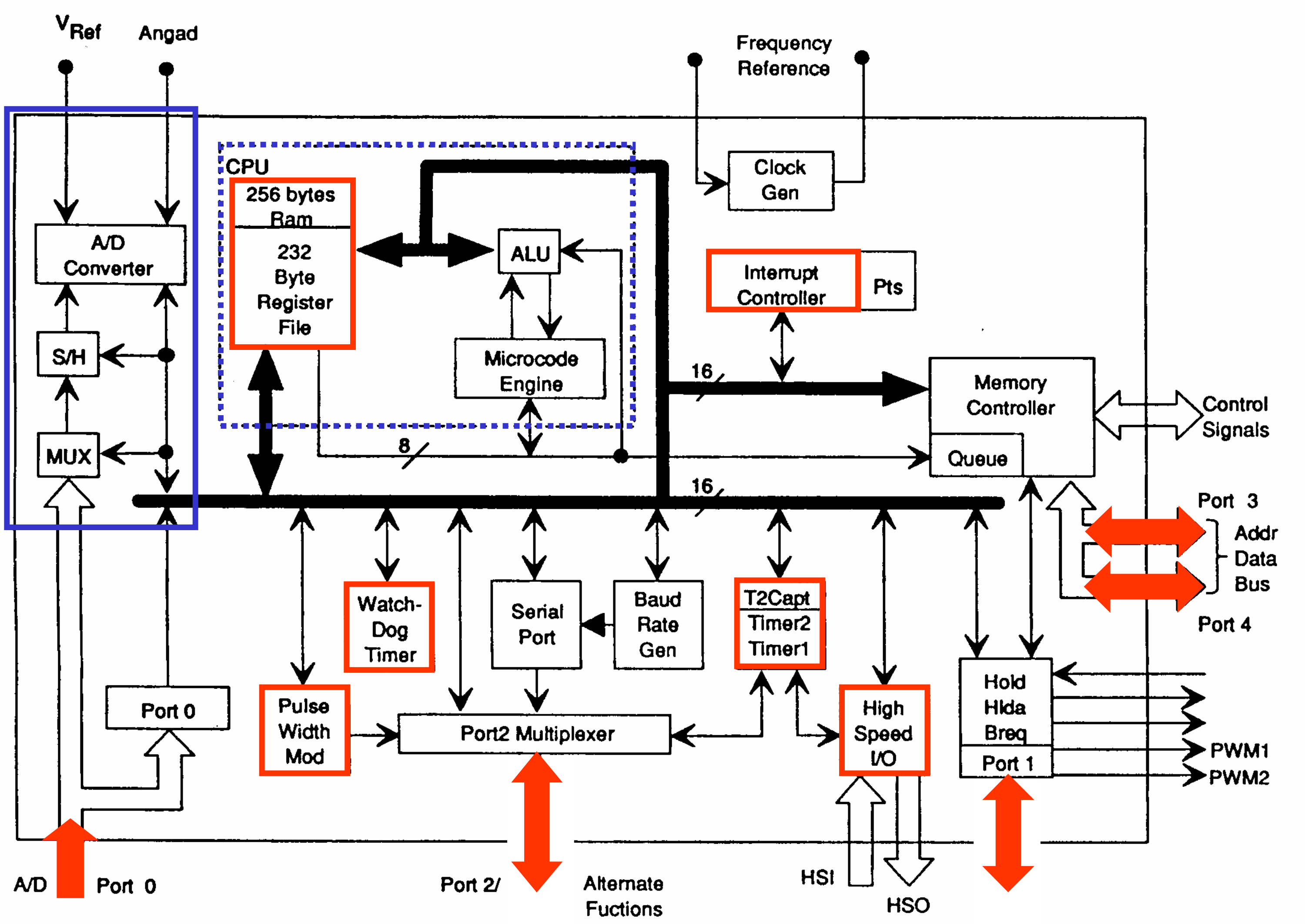
✓ Microcontrollers are 8 or 16 bits, fixed point devices.

✓ DSP's are usually 16 or 32 (fixed or floating point) bits devices.

✓ DSP Controllers merge a DSP architecture with Microcontroller peripherals. So far they are 16 bits devices.

✓ DSP's implement all the instructions in hardware. The execution of any instruction takes a cycle (excepting those breaking the pipeline).

80C196KC Microcontroller: Block Diagram



80C196KC internal buses:

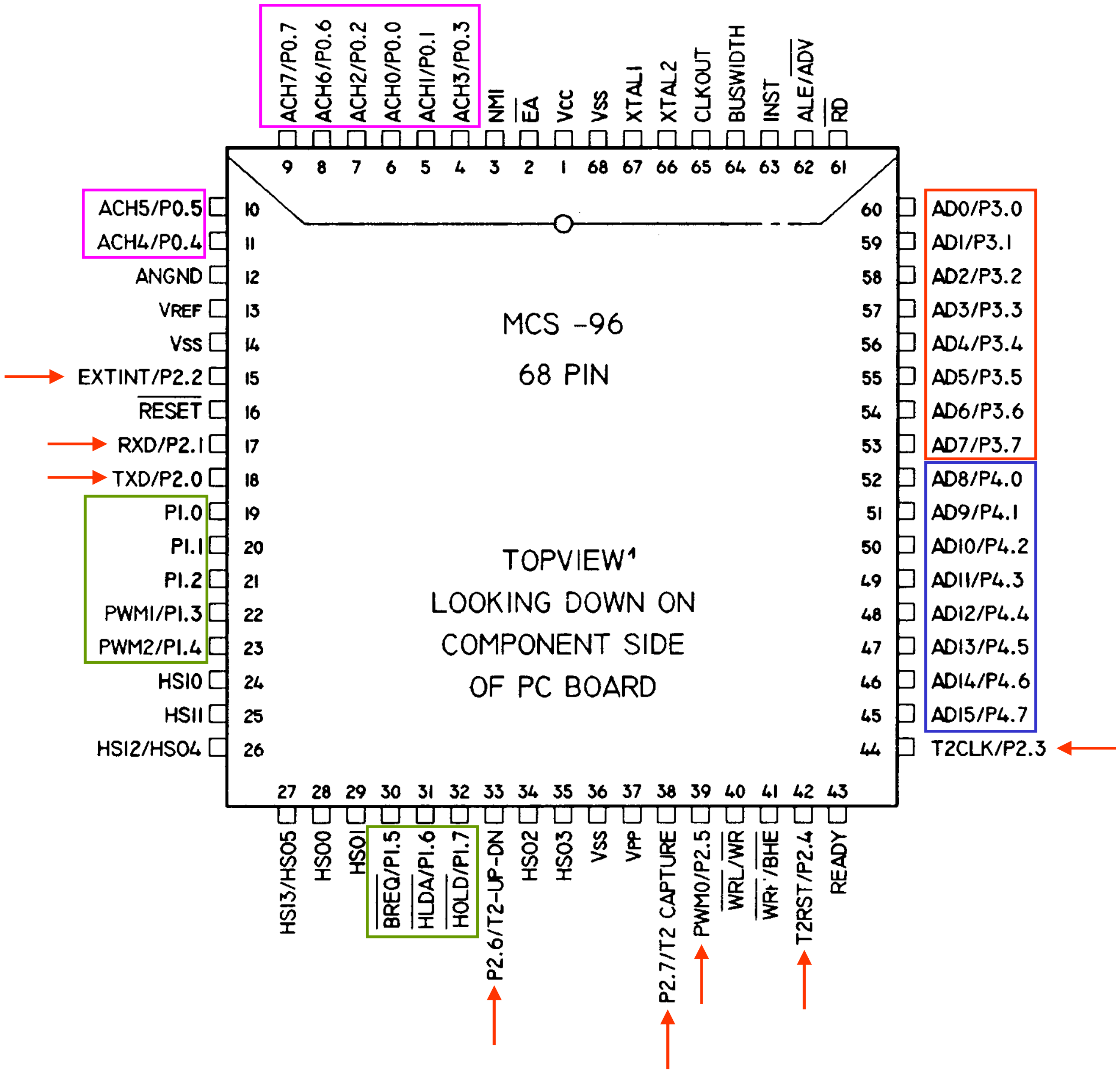
- 16 bits data, 8 bits address

80C196KC external buses

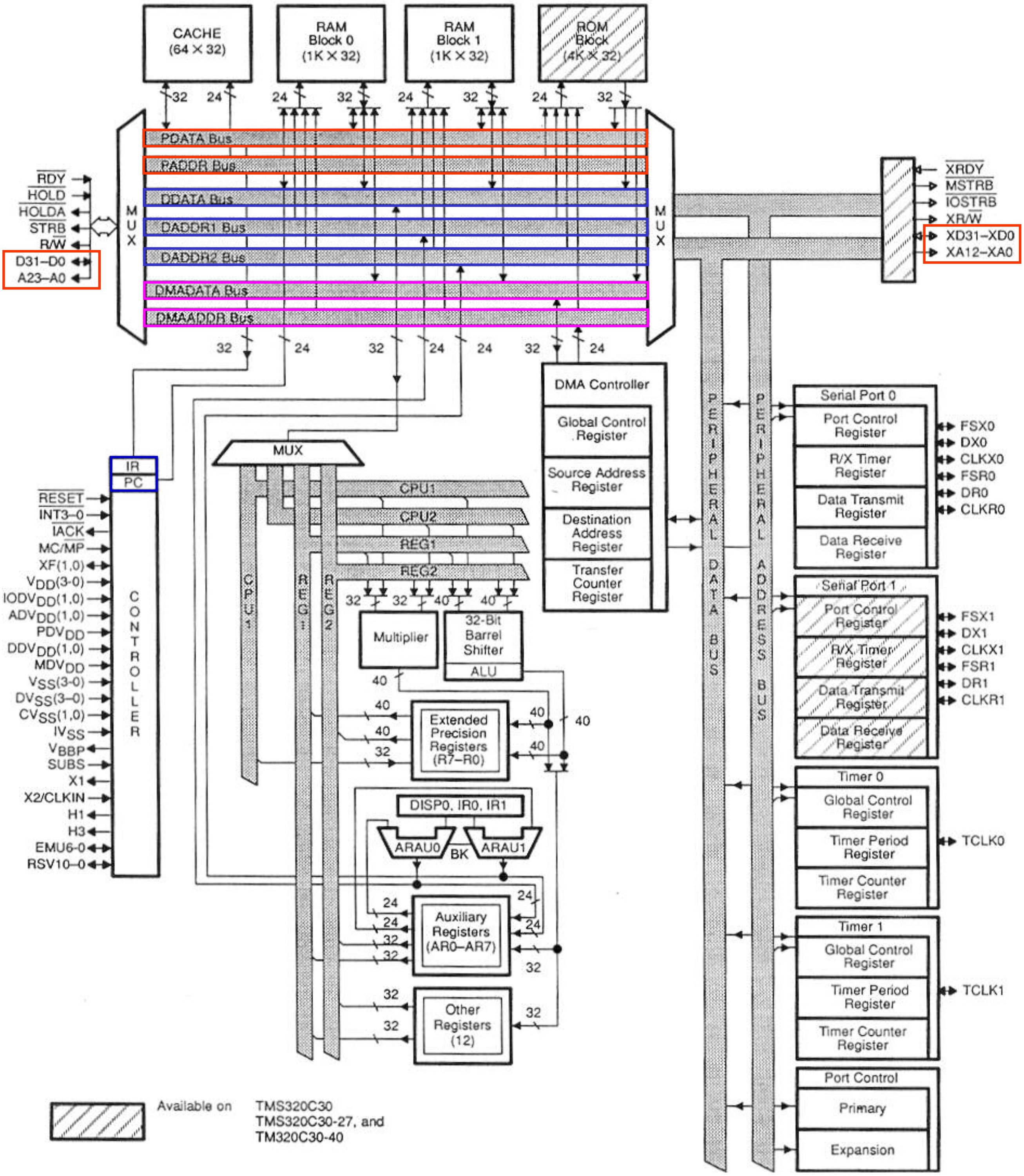
- Five 8-bits ports (0 to 4).
- Port 0: Input port, shares A/D converter inputs.
- Port 1: Bidirectional port.
- Port 2: Multi-functional port.
- Port 3 and 4: Bidirectional ports, addressed by the memory controller.

✓ Instructions require between 1 and 5 bytes.
✓ Can take from 2 cycles (e.g. EI) up to 38 cycles (e.g. DIV).

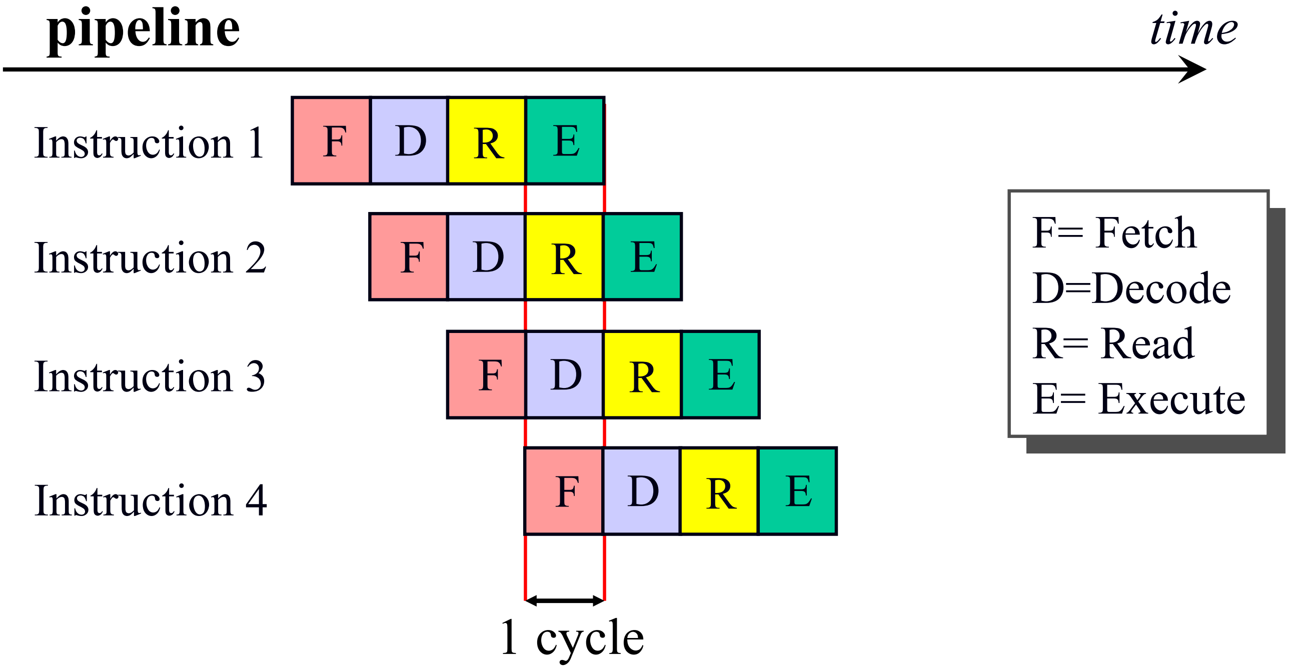
80C196KC Microcontroller: Pinout



TI TMS320C30 DSP: Block Diagram



TI TMS320C30-50 DSP (50 Mz clock): Performance



pipeline *time*

Instruction 1 F D R E

Instruction 2 F D R E

Instruction 3 F D R E

Instruction 4 F D R E

1 cycle

F= Fetch
D=Decode
R= Read
E= Execute

- ✓ Cycle instruction execution time = 2/50e6 = 40 ns.
- ✓ All instructions, excepting those breaking the pipeline, will take one cycle.
- ✓ Performs up to 25 MIPS (Million instructions per second).
- ✓ Performs up to 50 MFLOPS (Million-floating point operations per second).

Floating-point instructions (implemented on hardware)

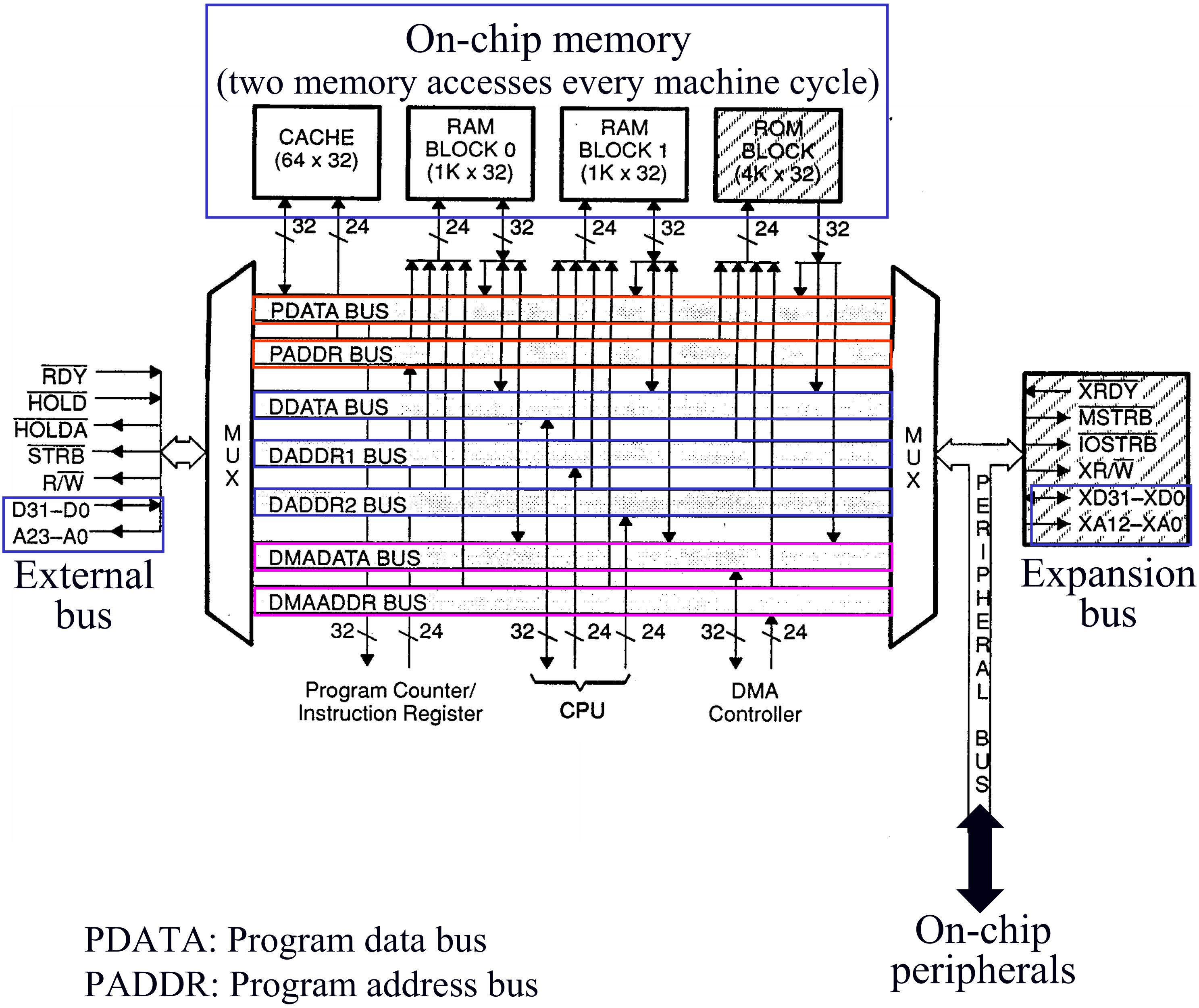
- ✓ Addition
- ✓ Subtraction
- ✓ Multiplication
- ✓ Absolute value

} 1 word (32 bits)
1 cycle

Some floating-point functions (implemented by software)

- ✓ Division: 32 words, 35 cycles (plus “call” and “ret”).
- ✓ Square root: 39 words, 35 cycles (average, iterative algorithms).

TI TMS320C30 DSP: Buses



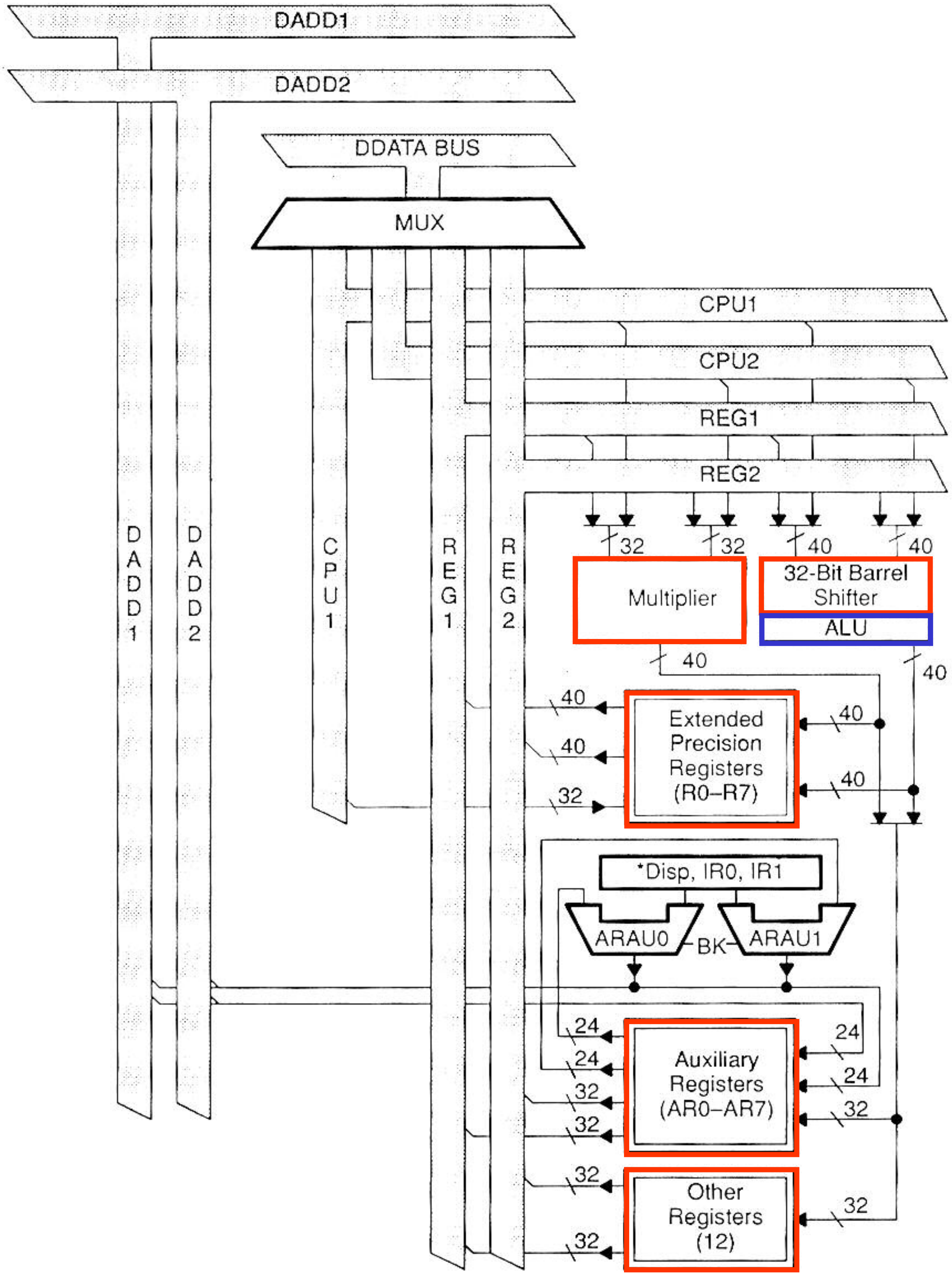
PDATA: Program data bus
PADDR: Program address bus

DADATA: Data data bus
DADDR: Data address bus (2) } Two data memory accesses every machine cycle

DMADATA: DMA data bus
DMAADDR: DMA address bus

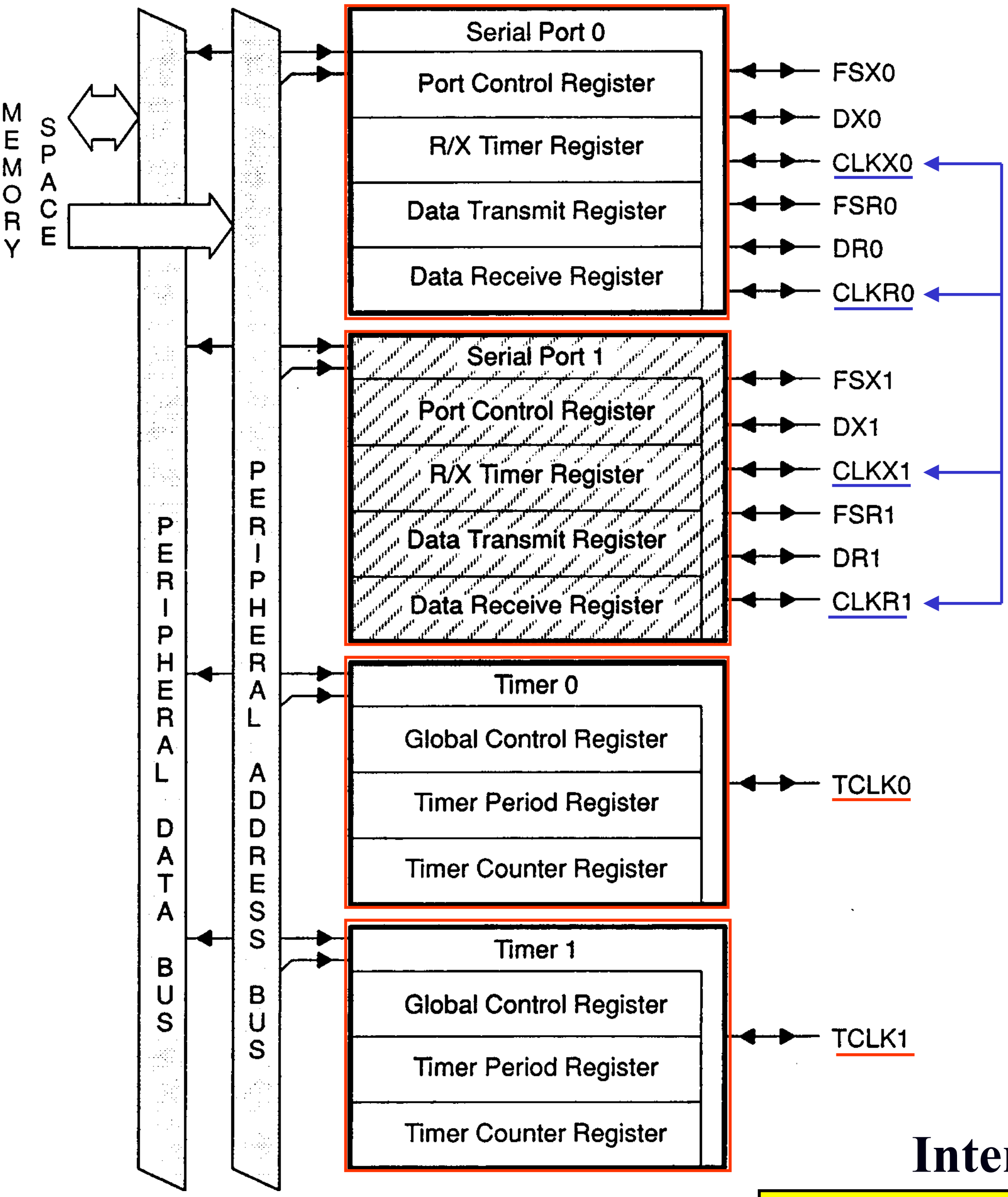
✓ Expansion and External buses allow one access every machine cycle

TI TMS320C30 DSP: CPU



ALU: addition, subtraction, absolute value, logical operations, etc
Barrel shifter: shifting, rotating, etc
Multiplier: multiplication

TI TMS320C30 DSP: Peripherals



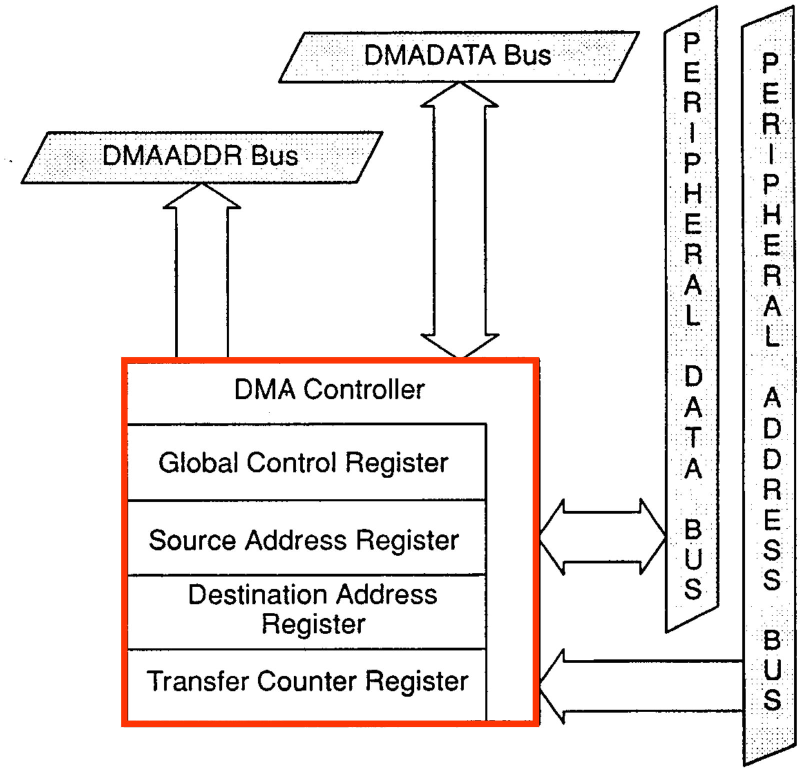
Available as 16 bits, general purpose timers when not needed for serial communications

Interrupts

Transmission serial port 0
Reception serial port 0
Transmission serial port 1
Reception serial port 1
Timer 0
Timer 1

TI TMS320C30 DSP: Direct Memory Access (DMA)

- Can read from or write to any location in the memory map without interfering with the operation of the CPU.
- The DMA can interface to slow external memories and peripherals without reducing the throughput to the CPU.



Programming Languages

	Efficiency	Developing Time	Portability
Assembler	☺	☹	☹
High level languages (C, Pascal, Matlab, ...)	😐	😐	😐
Graphic programming (Simulink, ...)	☹	☺	☺

Some side comments

✓ Programming using assembler requires a good knowledge of the processor architecture.

✓ High level languages often fail fully exploiting all the digital processor capabilities (advanced addressing modes, parallel instructions, etc).

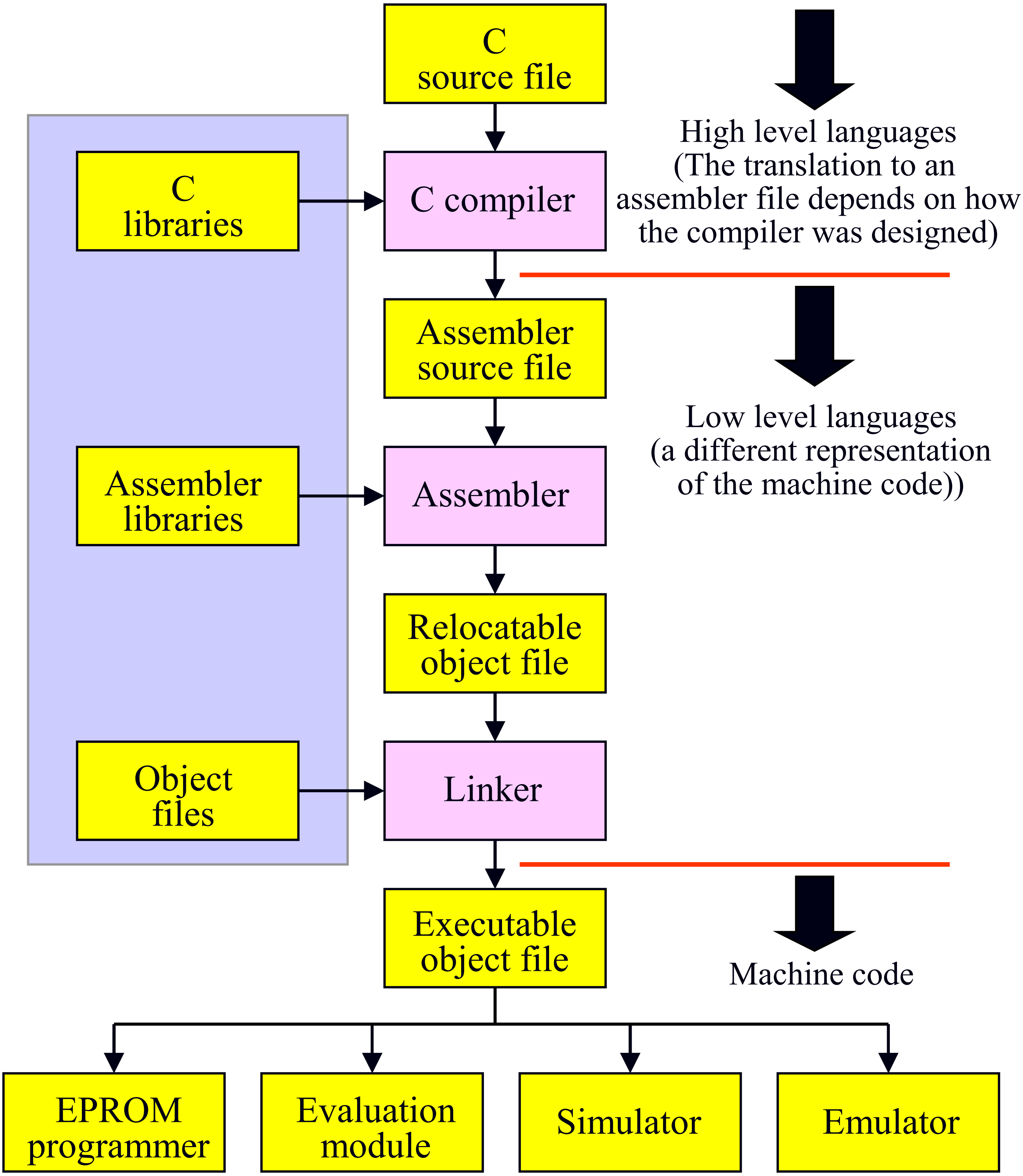
✓ Assembler functions are often used to program critical parts of the control algorithms.

✓ Digital processors manufacturers provide (free) optimized assembler code to execute frequently used algorithms (FFT, filters, ...).

✓ Portability of high level languages is limited as far as programs depend on the hardware configuration.

✓ High level languages (usually C) is more suitable when many programmers will be working together.

Programming Languages: Developing a Program Using C


C source file
C libraries
C compiler
High level languages (The translation to an assembler file depends on how the compiler was designed)
Assembler source file
Low level languages (a different representation of the machine code))
Assembler libraries
Assembler
Relocatable object file
Object files
Linker
Executable object file
Machine code
EPROM programmer
Evaluation module
Simulator
Emulator