MONASH University
Information Technology

FIT3159
Computer Architecture

Basic Machine Organisation / Systems and Busses

Dr Carlo Kopp, SMIEEE, SMAIAA, PEng
Carlo.Kopp@monash.edu
Clayton School of Information Technology,
Monash University
Australia

Why Study Basic Machine Organisation?

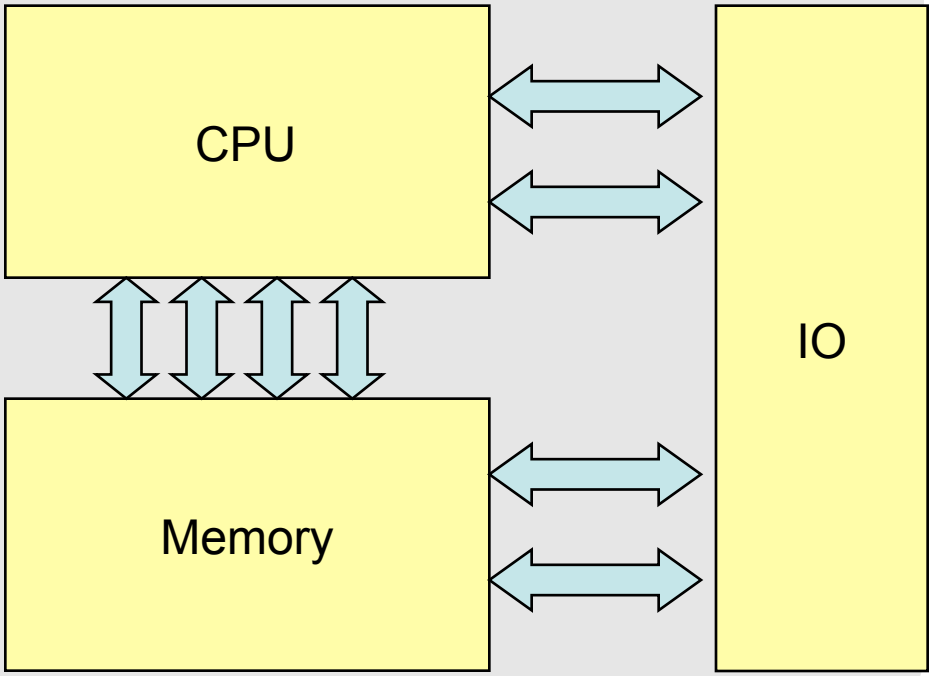
- Understanding how the internal components of a CPU core interact with each other – foundation knowledge.
- Appreciate the complexity of processor designs.
- Appreciate the impact on performance of various design strategies.
- Appreciate the different functions performed by different components in a computer system and within the CPU core itself.
- Understand that different machine organisations exist, with different design features to accommodate different needs in performance and functions.
- Prerequisite understanding for the study of individual machine components later in this unit.

Why Study Bussing?

- All modern machines employ a number of different internal busses for interconnecting internal components within the CPU cores, interconnecting cores, interconnecting memory, and interconnecting I/O.
- The study of bussing provides:
- Appreciation of the importance of interconnections between internal components in a machine.
- Appreciation of the impact of interconnections on machine performance.
- Appreciation of the flow of operands and instructions through the machine.
- Understanding of differences in various interconnection techniques and bus designs which is a prerequisite for understanding material in later lectures.

Basic Machine Organisation

- **Central Processing Unit**
  - Arithmetic Logic Unit (ALU)
  - Control Unit
  - Register Files
- **Main Memory**
  - Memory Controller
  - Memory Arrays
- **Input/Output (I/O)**
  - I/O Controllers
  - I/O Devices



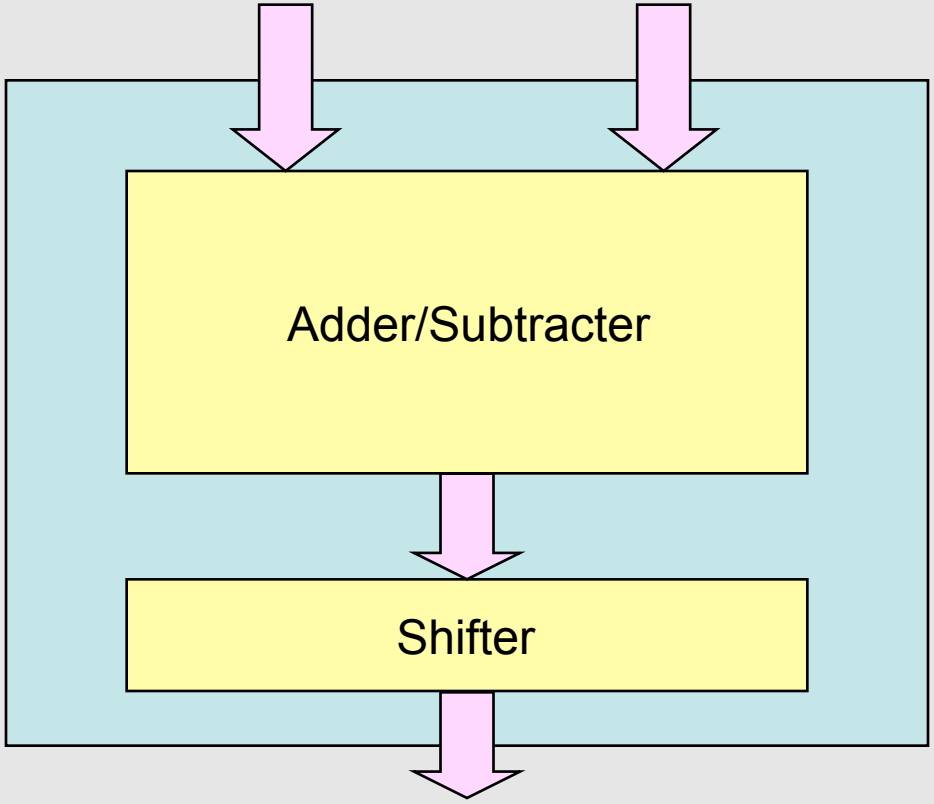
ALU Functions

The ALU is used to perform integer arithmetic operations such as:

- Addition
- Subtraction
- Shift Left / Shift Right
- Integer Multiply and Divide
- Addressing Calculations
  - > Used for addressing complex structures like arrays
- Logical and Arithmetic Tests
- Jump Address
  - > Used for control structures like case statement

ALU Structure

- **Usually handle Integer Add/Subtract and Logical functions with same unit**

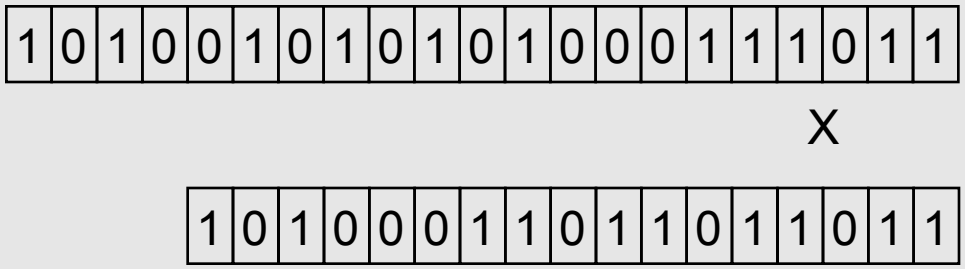


Fast Adders

- The preferred approach to designing adders is the use of the "Carry Look-Ahead" adder.
- Conventional "ripple adders" are slow because the value of the carry from each full adder must propagate through each and every stage in the adder before the addition can complete (refer tutorial). The time to complete the addition is then proportional to the number of bits in the addition operation.
- In a "Carry Look-Ahead" adder, of which there are several different types, combinatorial logic is used to calculate the carries before the sums are calculated.
- As a result a "Carry Look-Ahead" adder can compute the sum of two integers in a time which is not proportional to the number of bits in the two numbers being added.

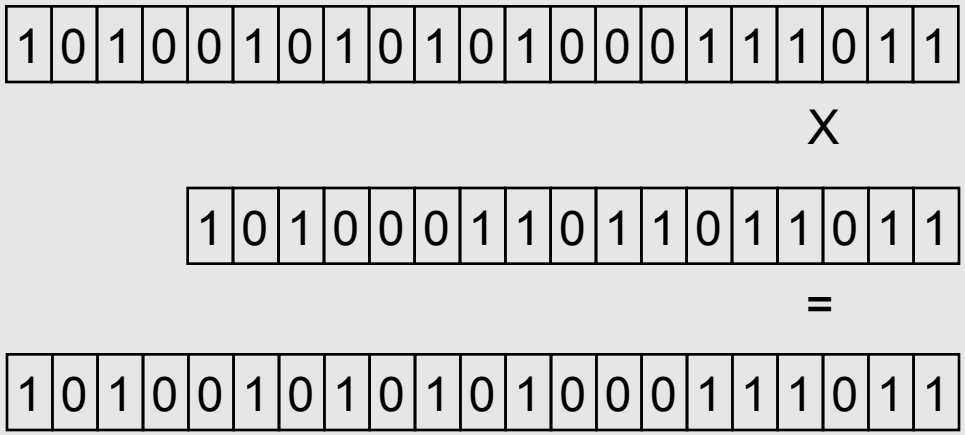
Multiplication/Division

- **Either done with**
  - Repeated add and shift operations (slow)
  - Wallace tree parallel multipliers (fast and widely used now)



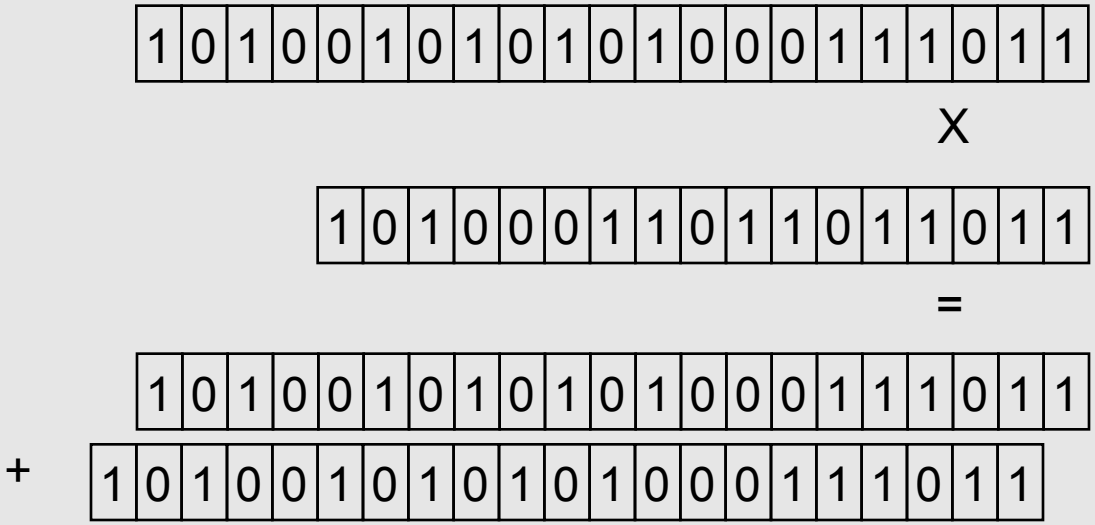
Multiplication/Division

- **Either done with**
  - Repeated add and shift operations
  - Wallace tree parallel multipliers



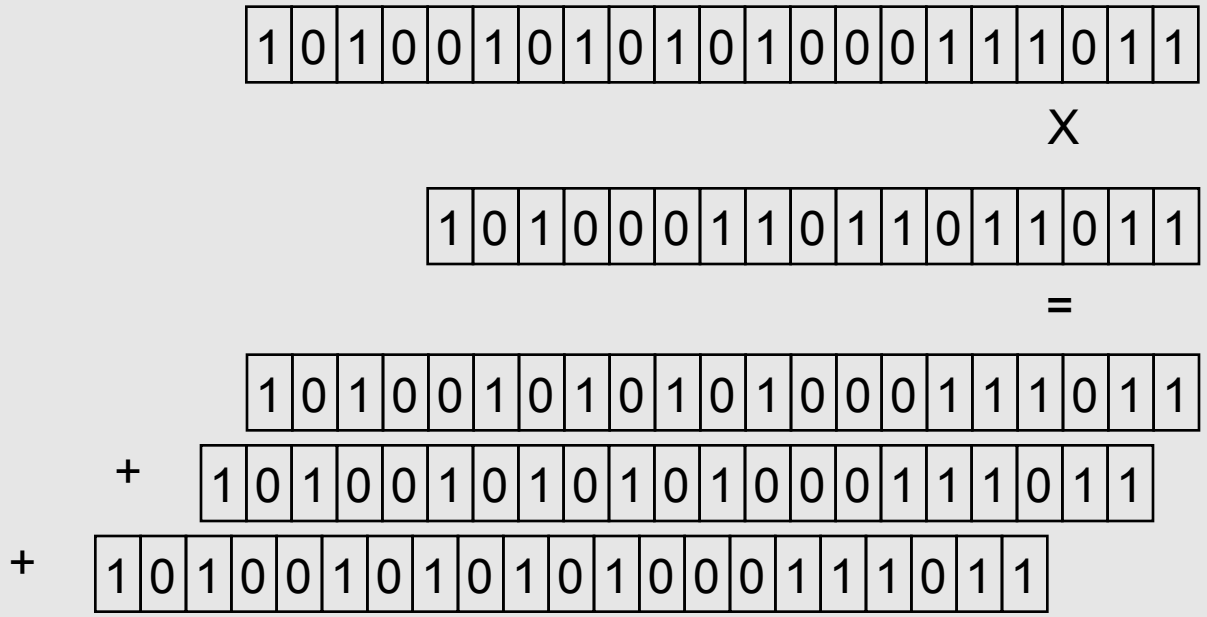
Multiplication/Division

- **Either done with**
  - Repeated add and shift operations
  - Wallace tree parallel multipliers



Multiplication/Division

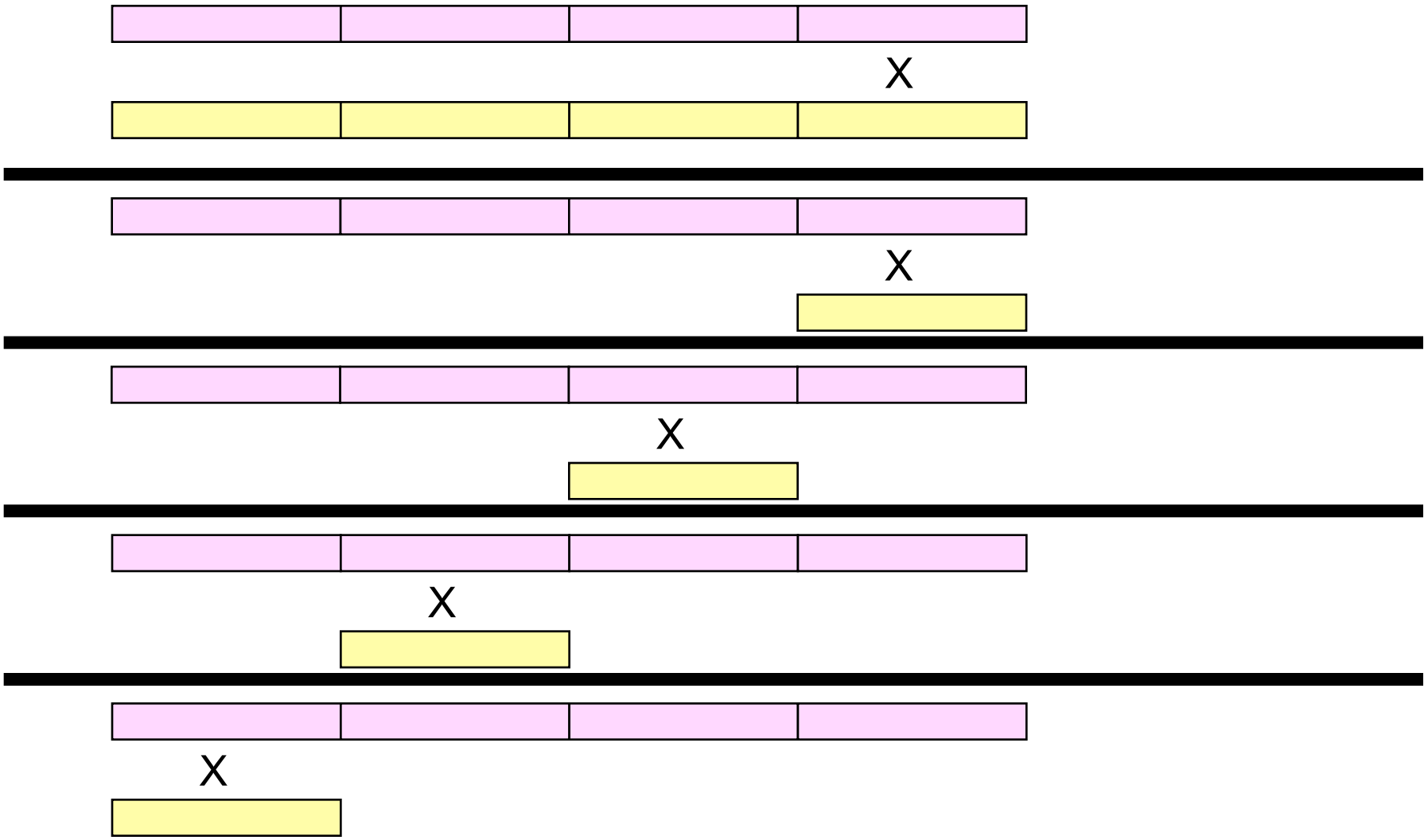
- **Either done with**
  - Repeated add and shift operations
  - Wallace tree parallel multipliers



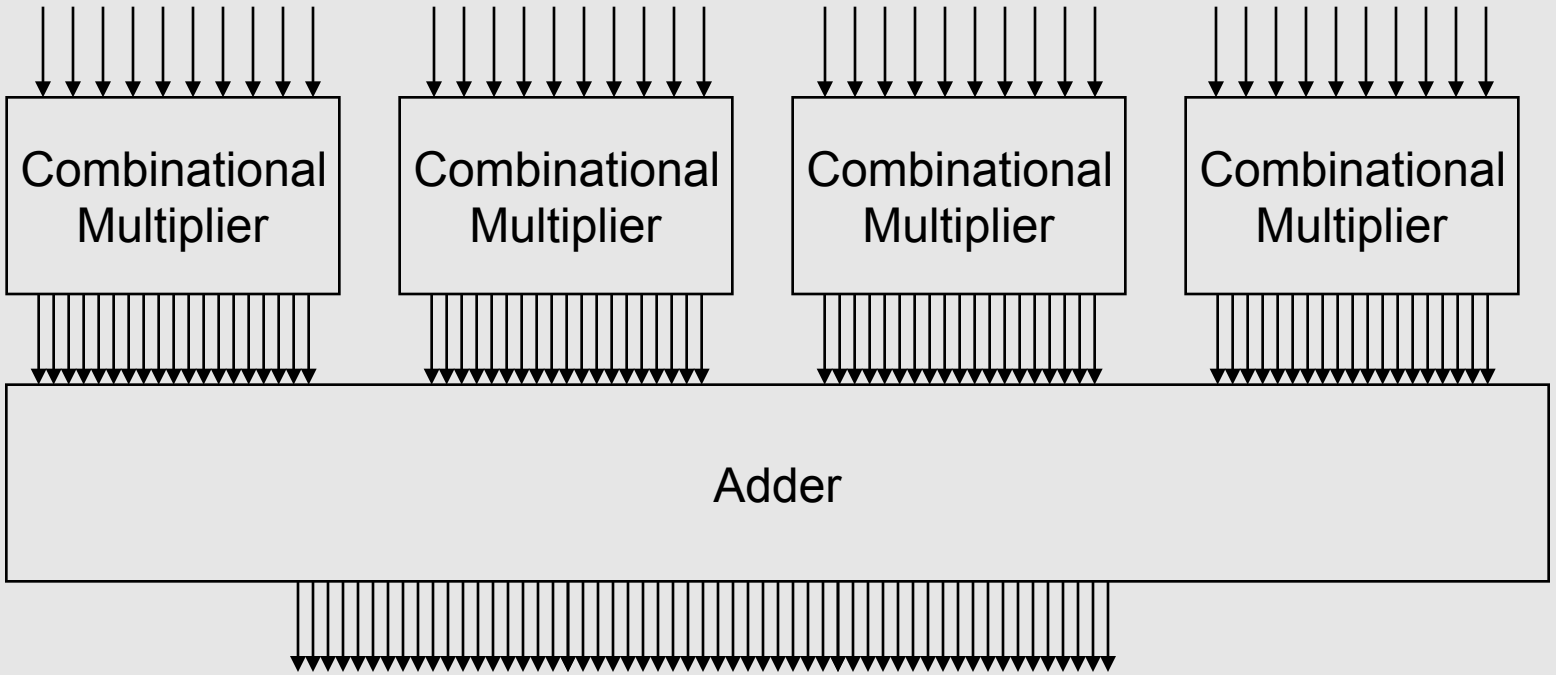
Tree multipliers

- Tree multipliers (eg “Wallace tree”) will multiply two integer numbers efficiently and much faster than repeated shift and add operations.
- This is performed using combinatorial logic.
- The principle of all such multipliers is to perform multiplication of two input numbers to produce partial products which are then summed to produce the full product.
- The “Wallace tree” multiplier uses a tree structure of modified adders to combine the partial products quickly and efficiently. This is done until all of the partial products have been added.
- Tree multipliers are now used in most modern microprocessor chips.
- 8-bit microprocessors still often use shift/add multiplication.

Tree multipliers

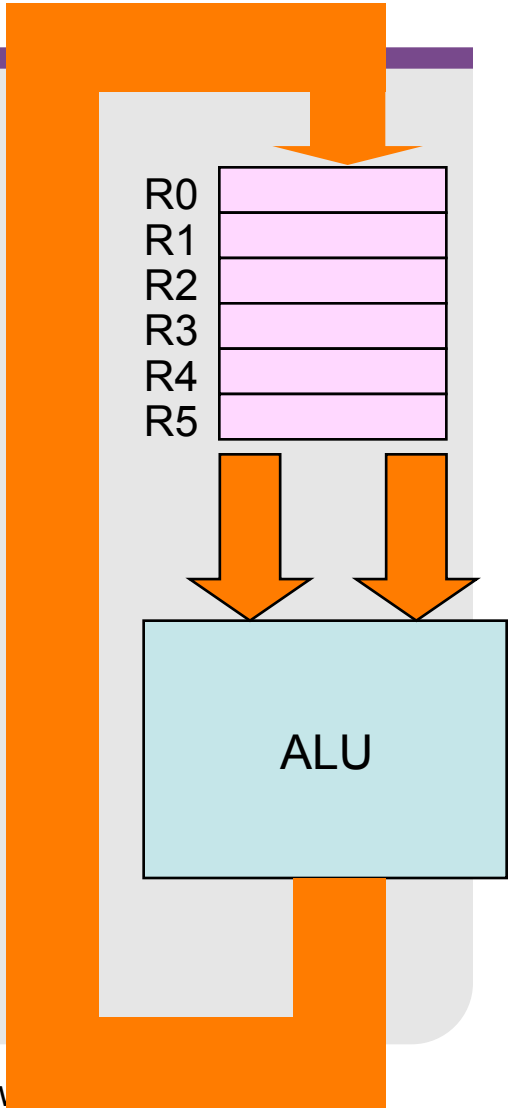


Tree Multipliers



Register File Functions

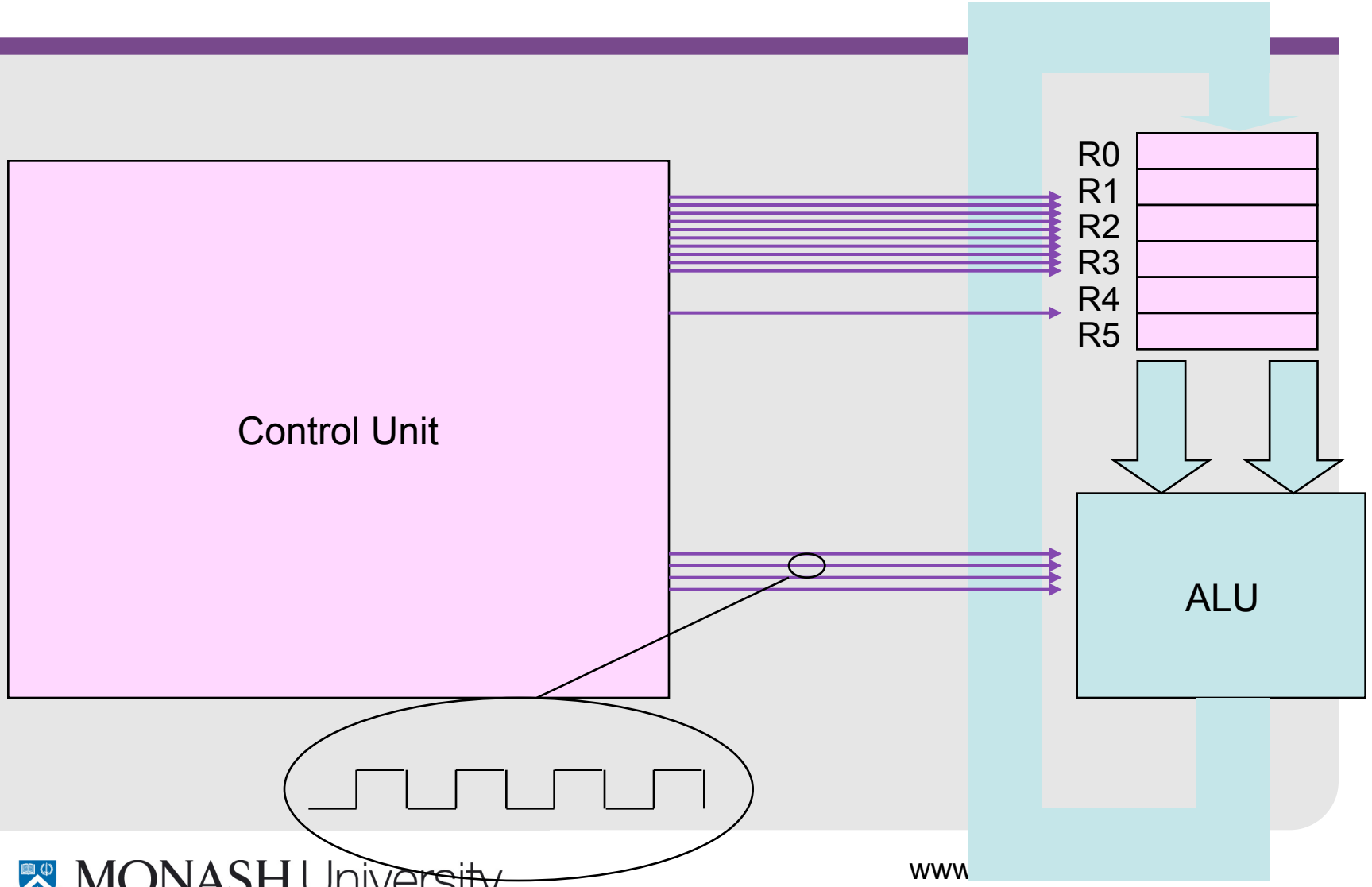
- The Register File is mostly used for the temporary storage of operands.
- The Register File contains a bank of registers, each typically of the native word size of the machine.
- A typical architecture will allow operands to be moved between registers, and also loaded from and stored to the Main Memory and I/O.
- Register File sizes vary between several registers, up to potentially dozens or hundreds, depending on the architecture.
- Usually need multiple ports on register file
  - At least 2 read ports and 1 write port per cycle



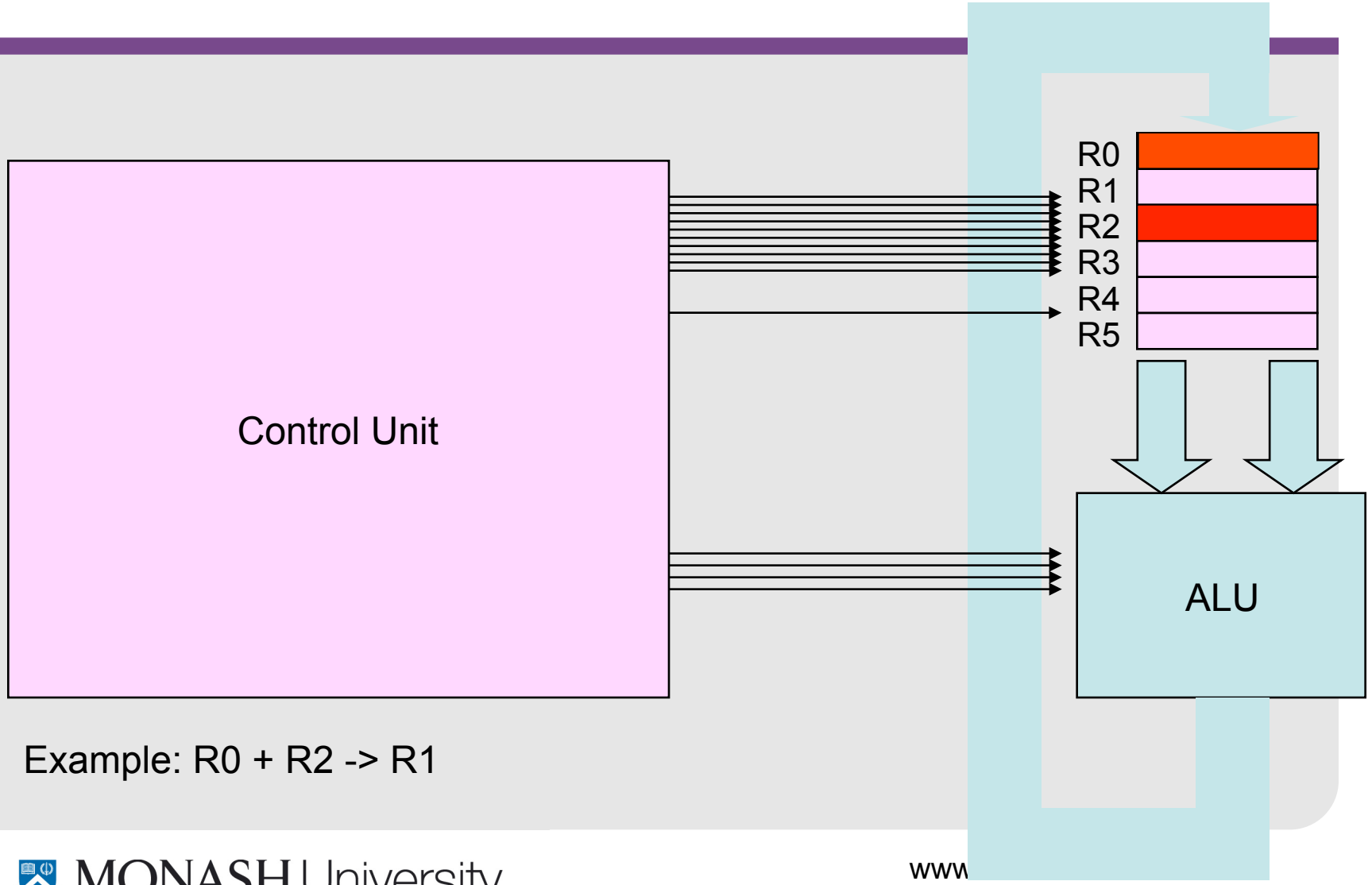
Control Unit Functions

- The Control Unit is employed to coordinate and control the operation of all components within the CPU.
- The Control Unit will decode machine instructions (ie the program) fetched from memory, in order to generate appropriate control signals to drive the CPU.
- Control Unit internal design is highly specific to particular machine architectures, or families of architectures.

Control Unit

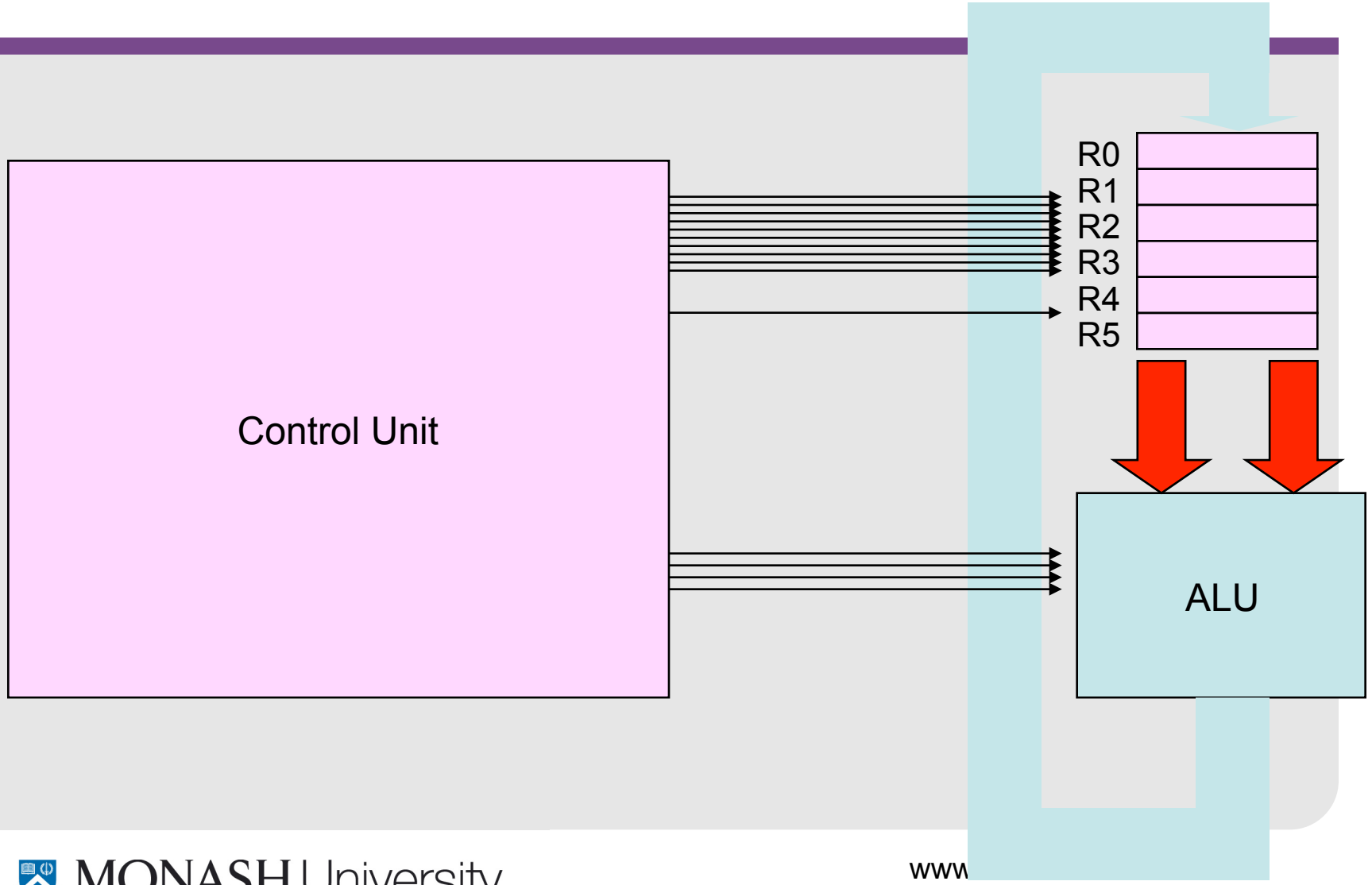


Control Unit

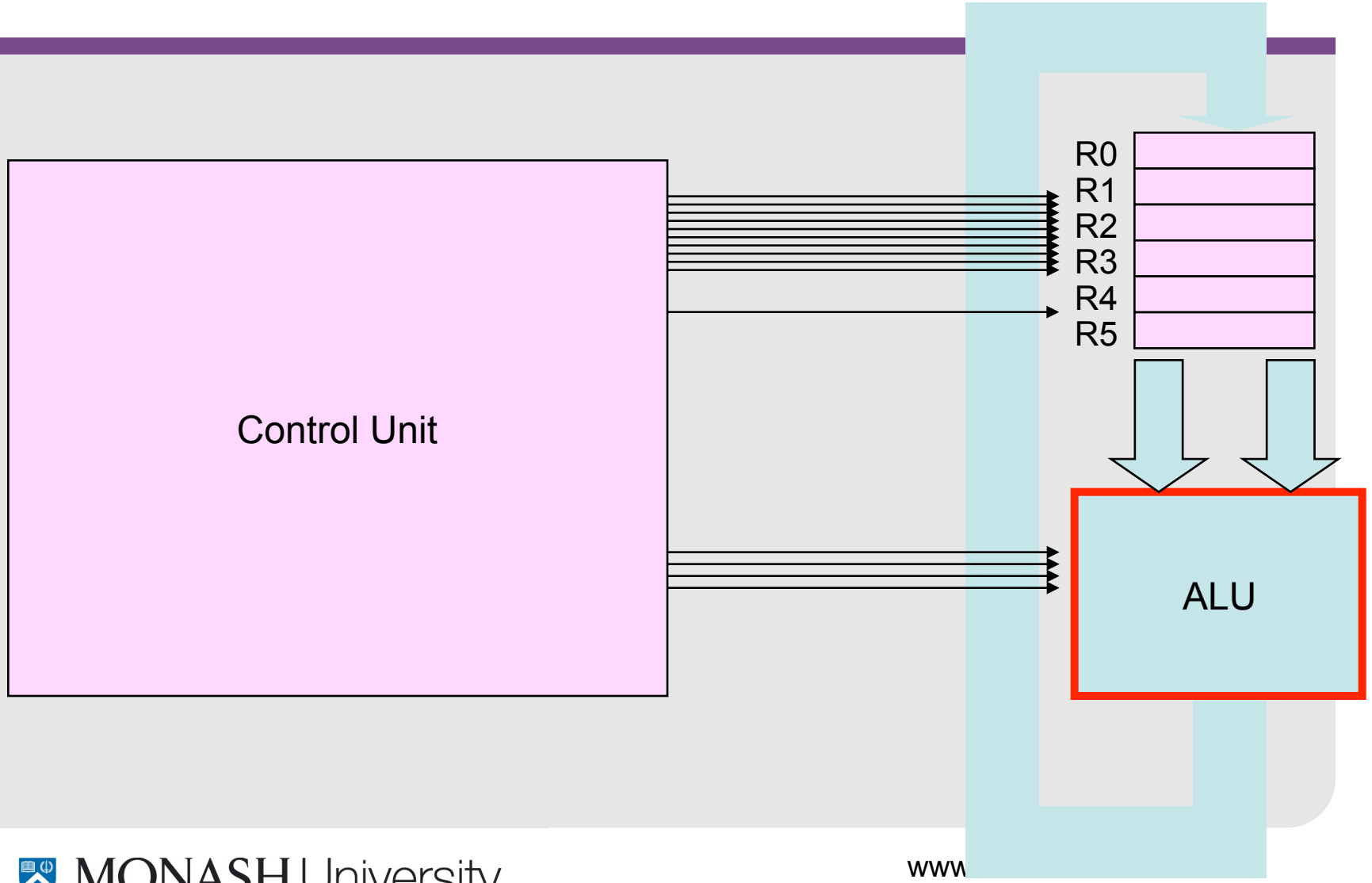


Example: R0 + R2 -> R1

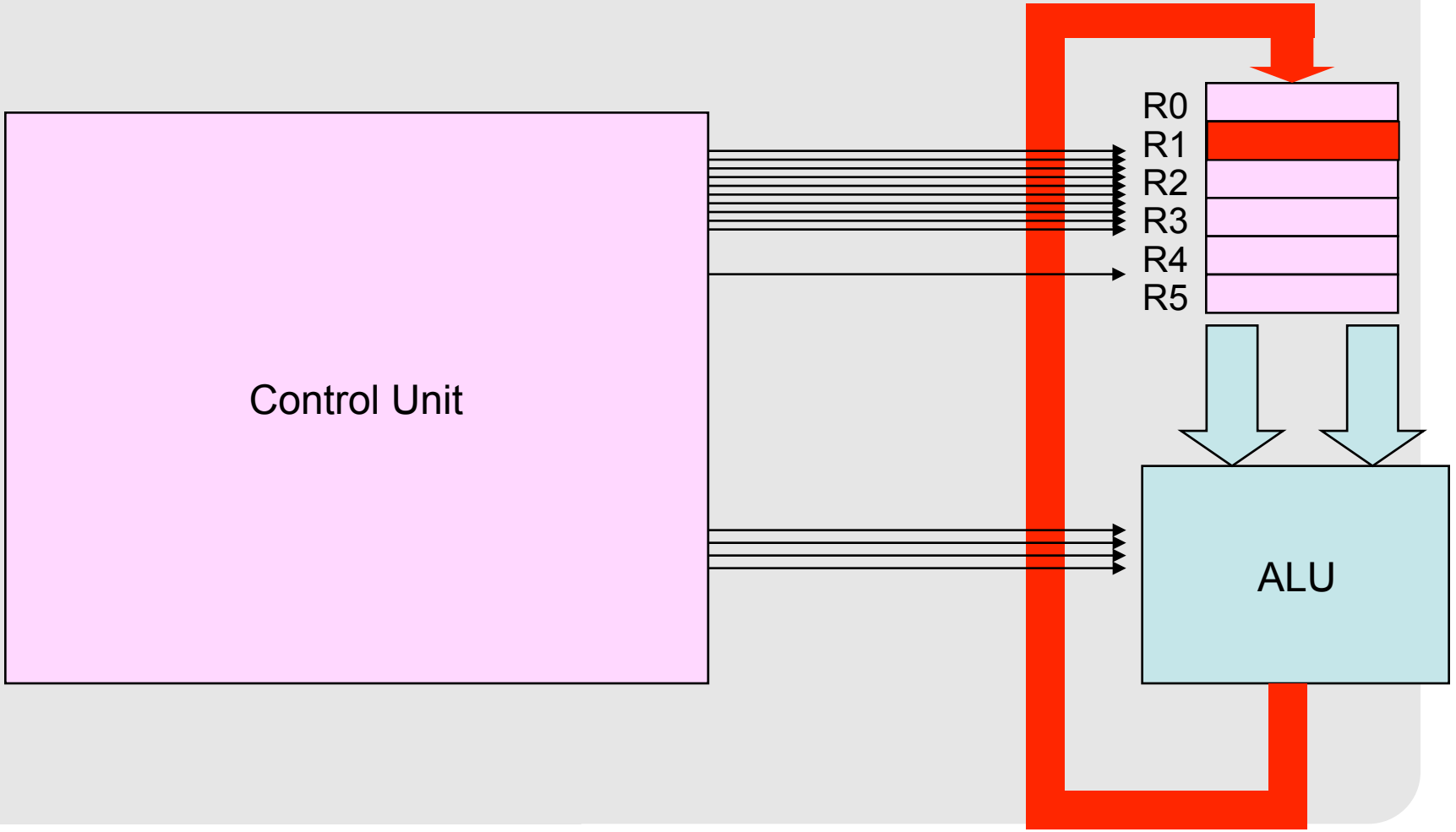
Control Unit



Control Unit



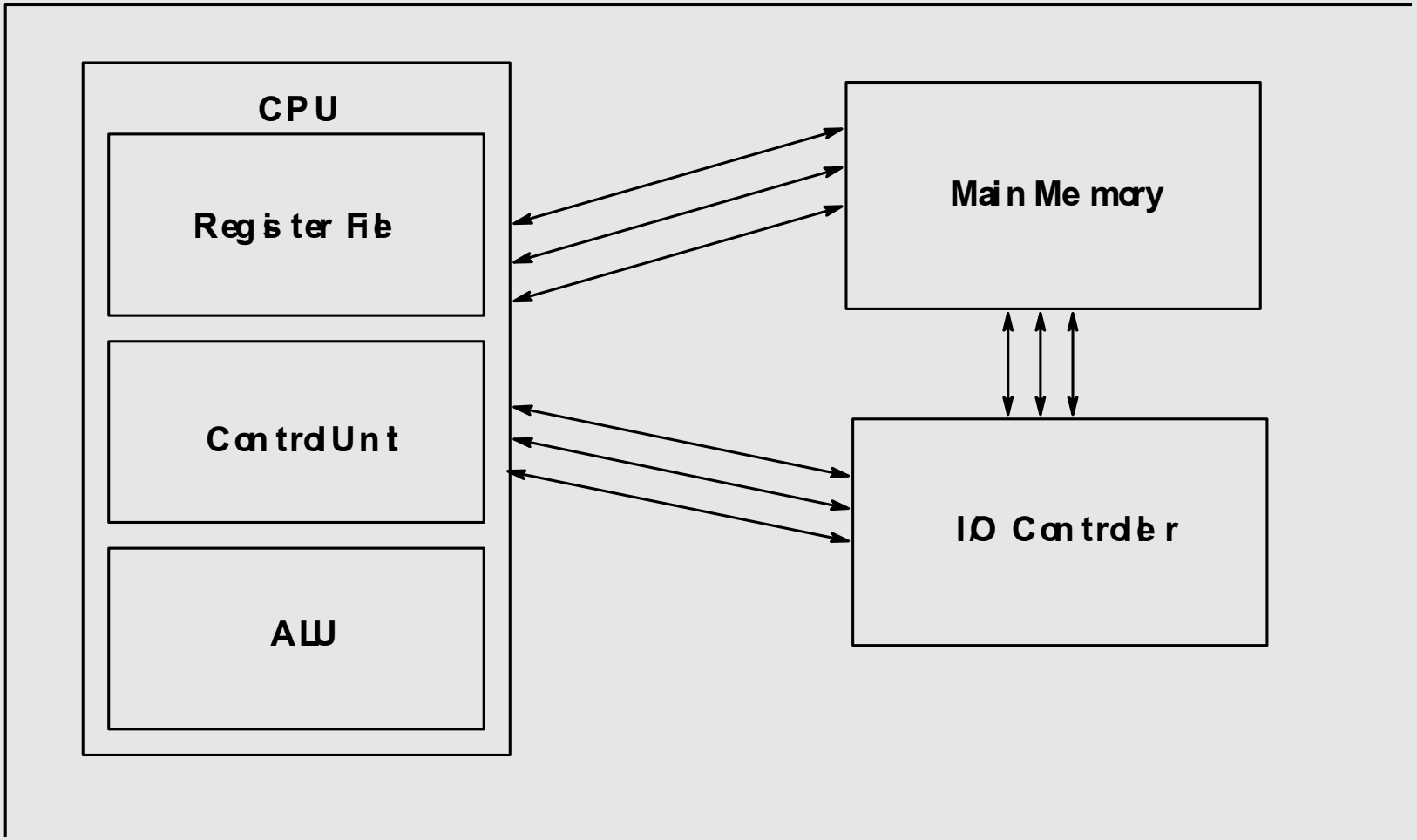
Control Unit



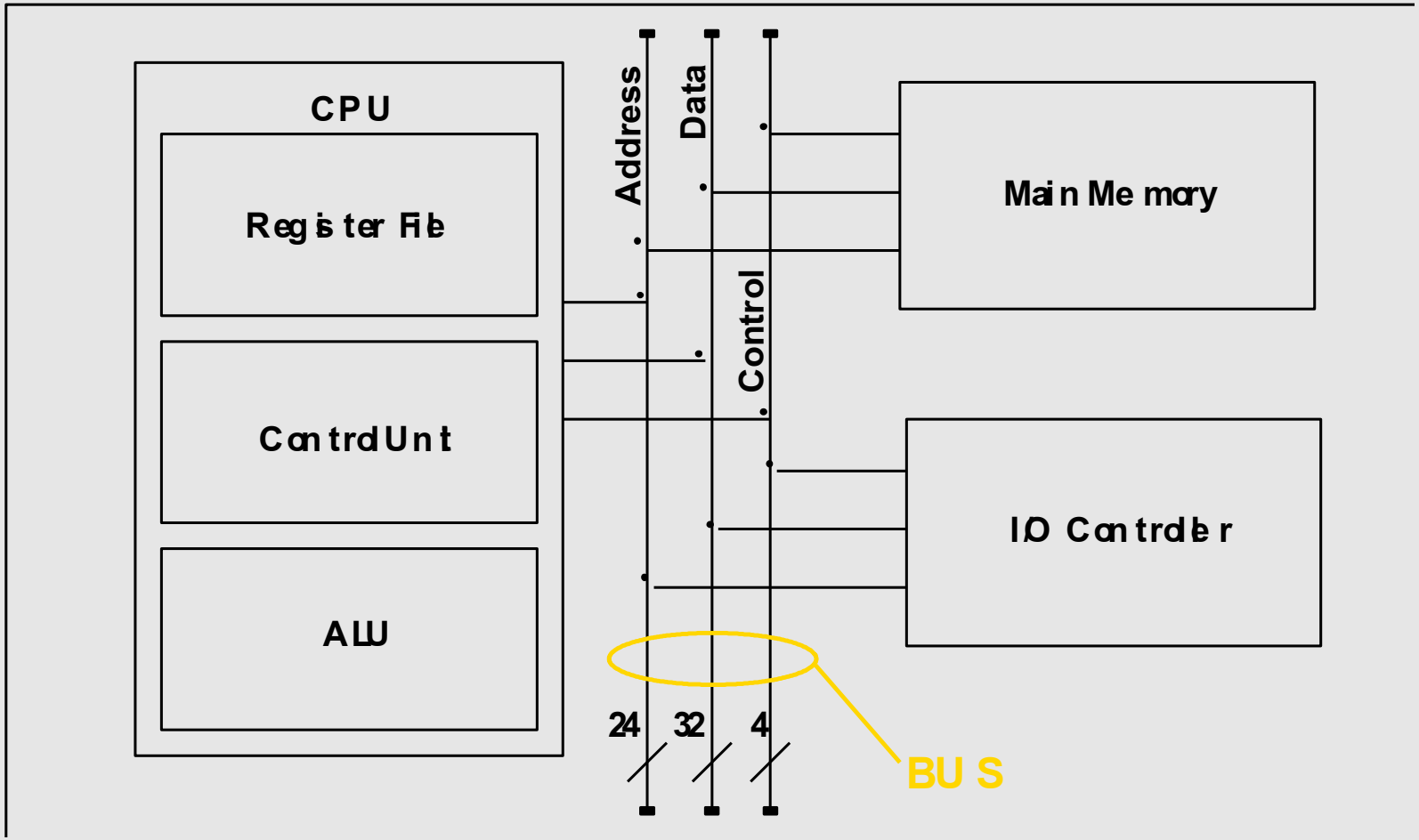
Interconnecting Computer Internals

- **"Spaghetti Strategy"**
  - Bundles of wires are used to implement specific data, address and control connections between the CPU, Memory and I/O, and between CPU components.
  - This approach is cumbersome, difficult to maintain, and not used in modern designs.
- **"Bussing Strategy"**
  - The components in the machine share, where appropriate, interconnecting data and address lines.
  - Most commonly used in modern computers.

Spaghetti

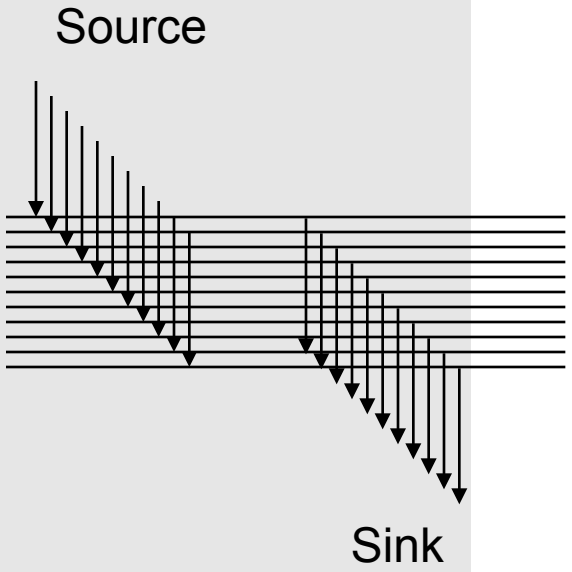

CPU
Register File
Control Unit
ALU
Main Memory
I/O Controller

Bussing


CPU
Register File
Control Unit
ALU
Address
Data
Control
Main Memory
I/O Controller
24
32
4
BUS

Bussing

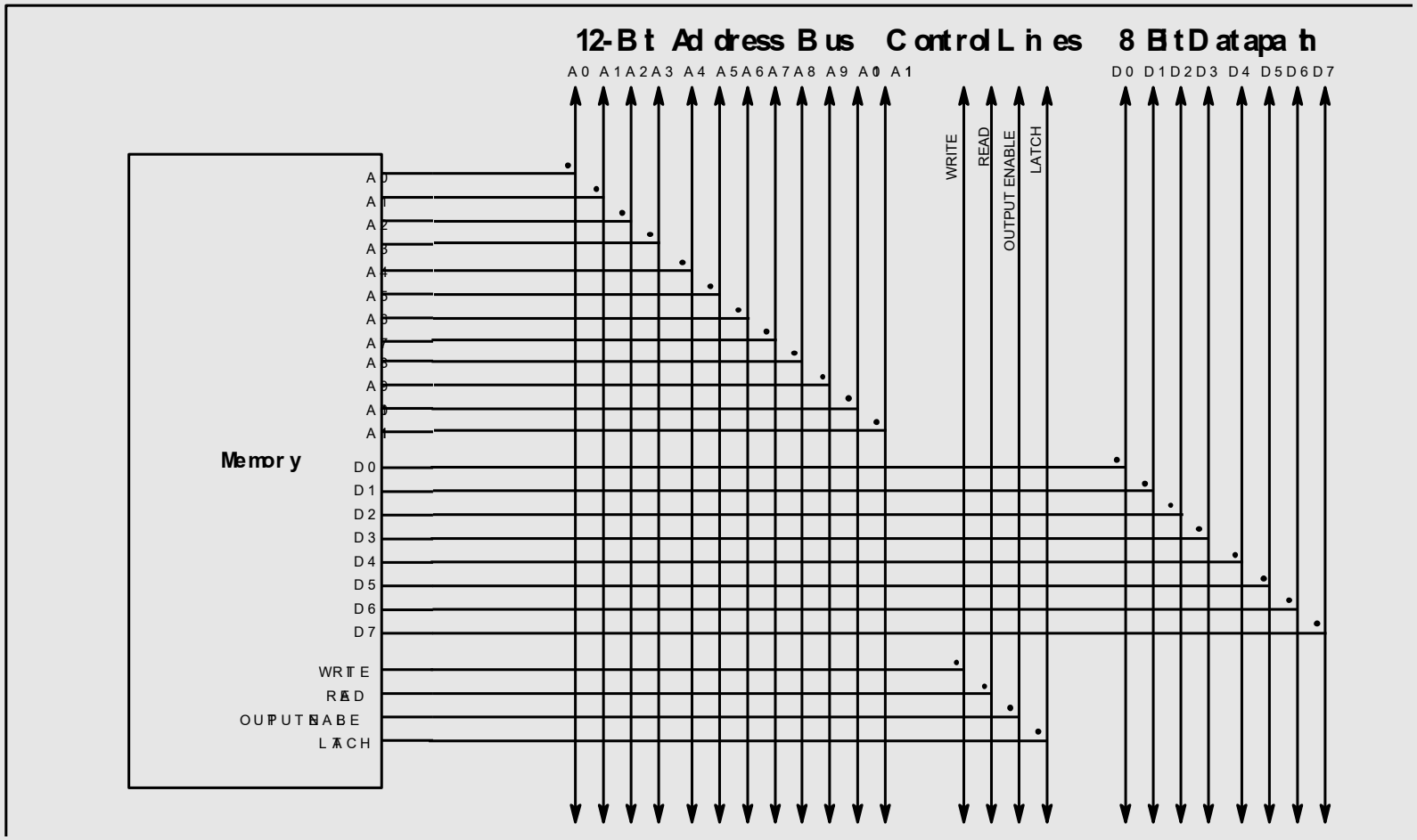
- A *bus* is, in the simplest of terms, a group of wires, within which each wire has a defined logical function, and which is employed to carry data, address and/or control information.
- Every *bus* is characterised by a "*bus protocol*", which is a specific set of rules for the transfer of data across the bus. Every device connected to the bus must comply with the protocol for the bus to perform properly.



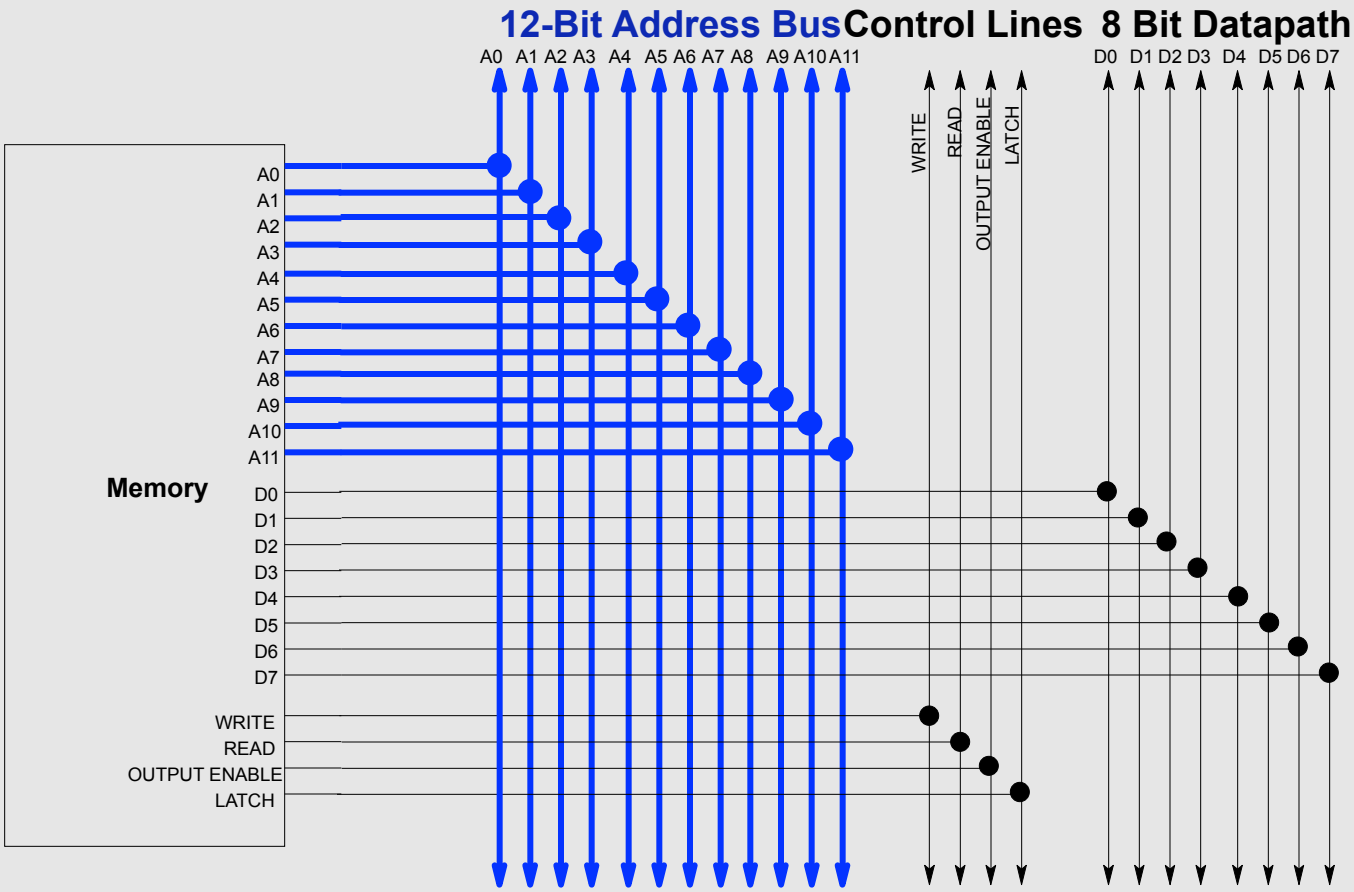
Bus Structure

- **It is customary to group the lines (wires) in a bus by their respective functions:**
  - *Address Bus* - a group of lines carrying only addressing information.
  - *Data Bus* - a group of lines carrying only data.
  - *Control Bus* - a group of control lines.
  - Some busses share *Data* and *Address Bus* lines to increase performance, or reduce costs.
- **Note that the address and control bus lines must be asserted before the data lines become active – whether we are reading or writing, the source or destination must first be addressed and type of transfer asserted before the data can be asserted to the data bus lines; this is true regardless of the type of bus involved.**

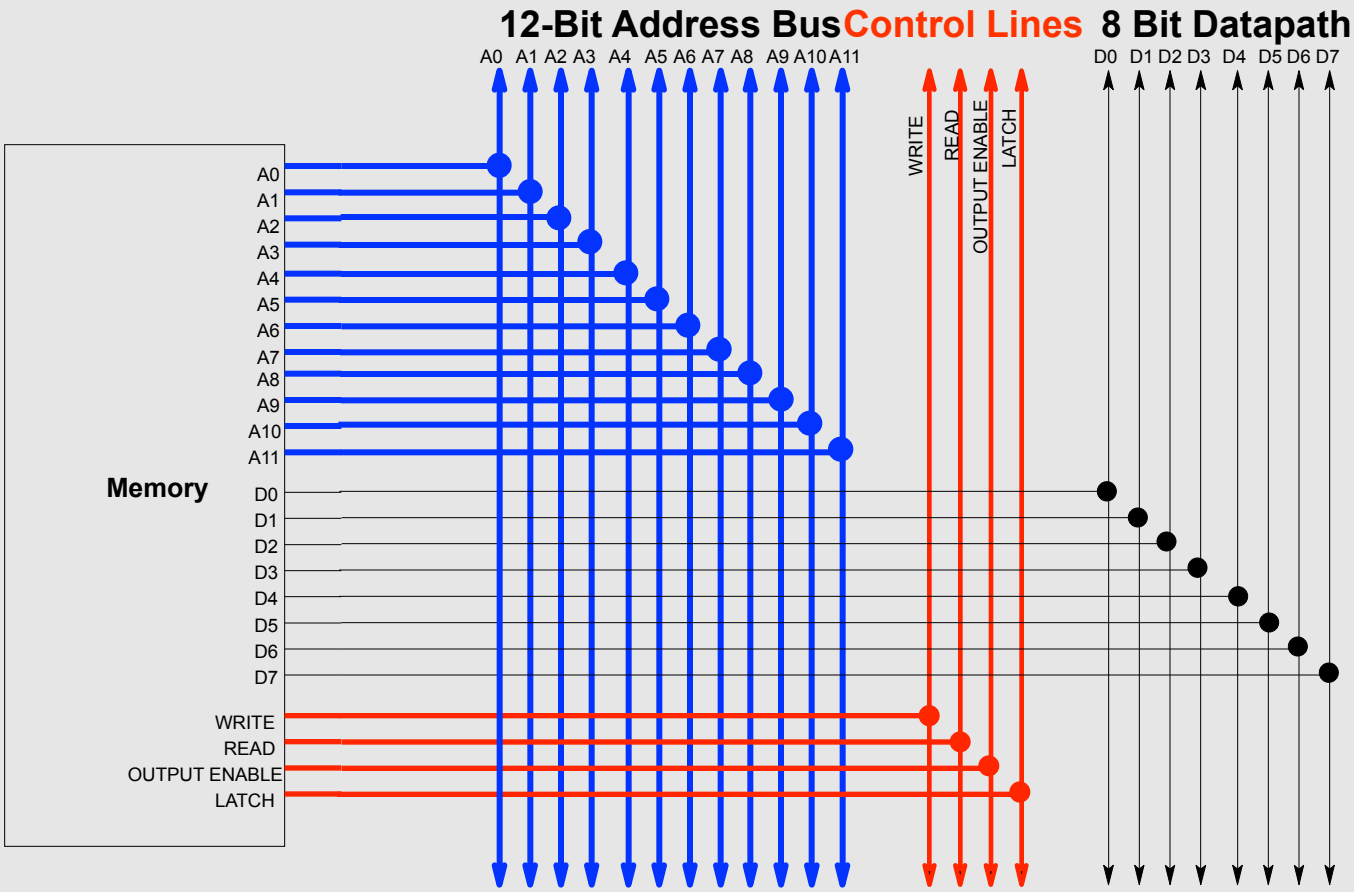
A Very Simple Bus Design



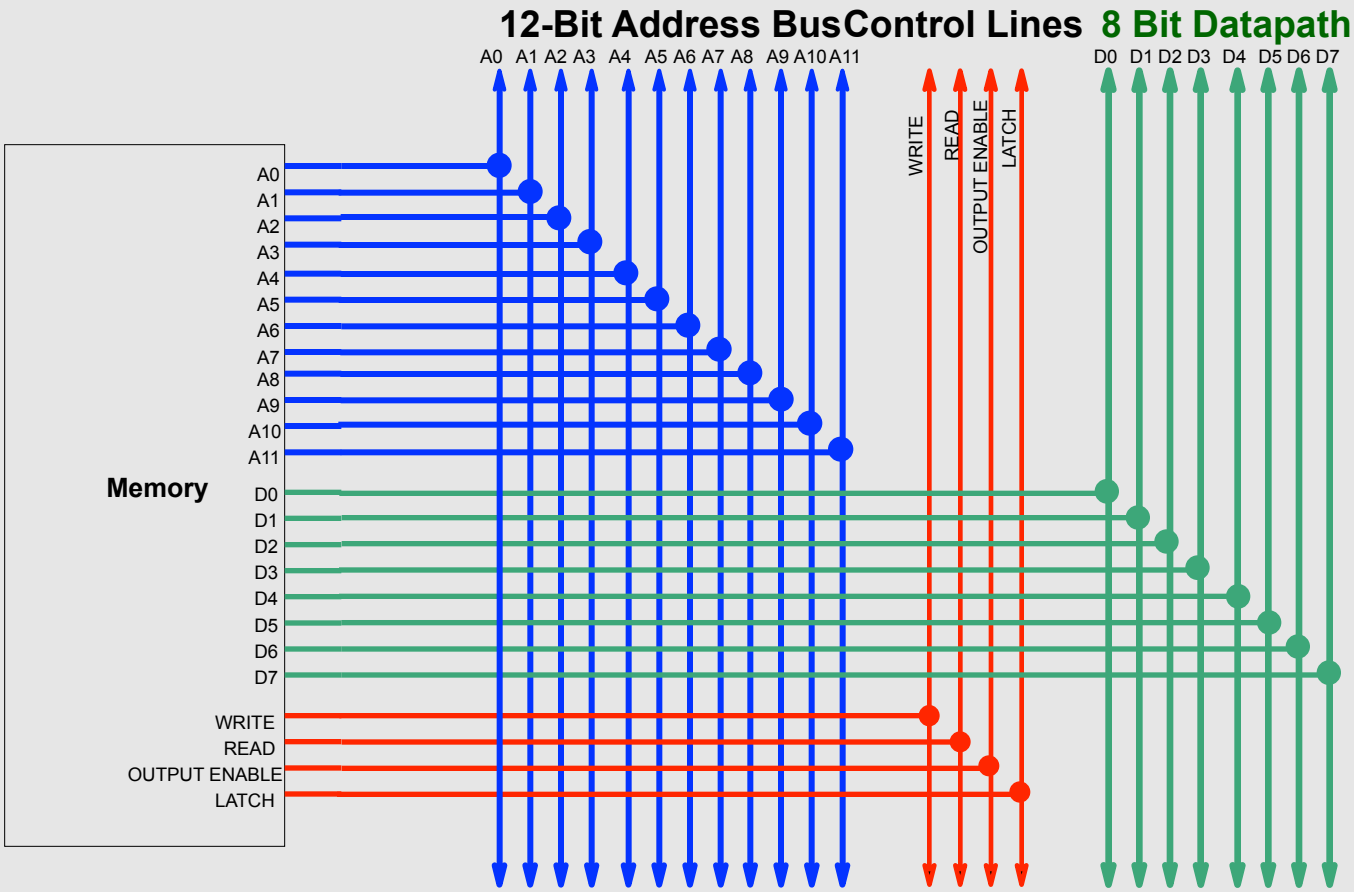
A Very Simple Bus Design



A Very Simple Bus Design



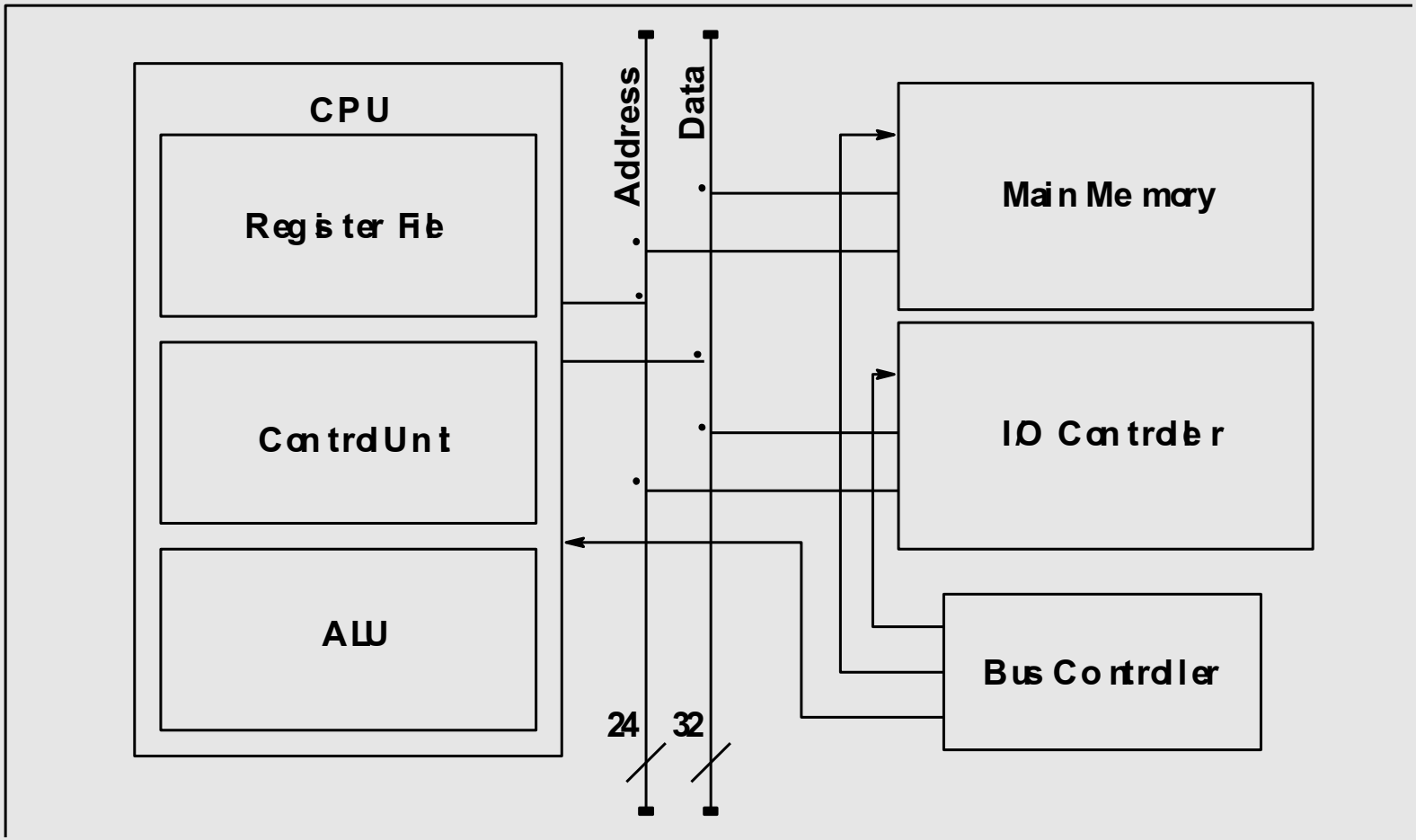
A Very Simple Bus Design



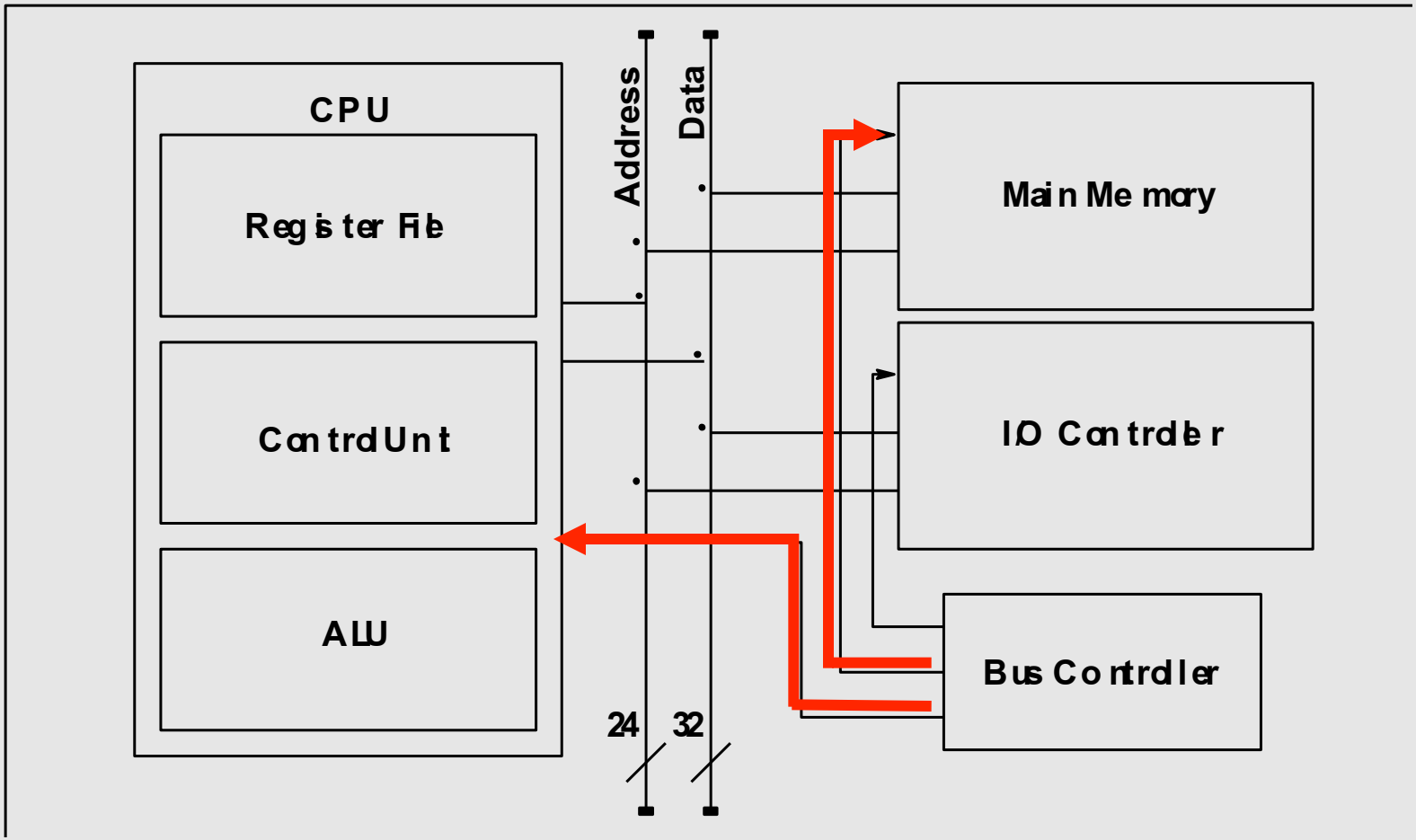
Bussing - Centrally Controlled

- The simplest type of bus is a centrally controlled bus.
- In such a bussing scheme, there exists a device called a “bus controller” which controls all transfers between other devices on the bus.
- The controller will issue commands such as “read” or “write” to devices on the bus, such as memory or I/O controllers. The devices are thus “slaves” to the controller, which is the “master”.

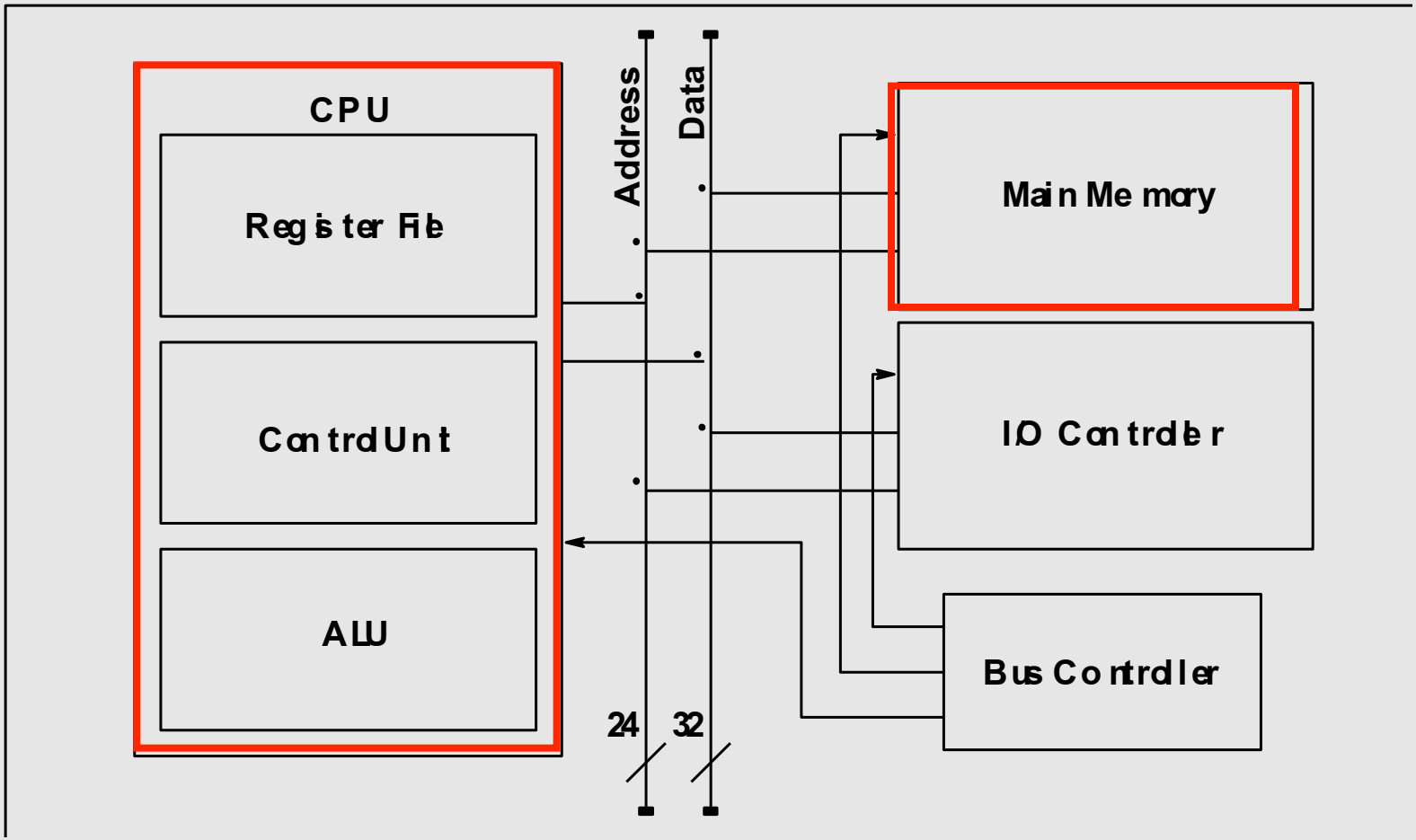
Centrally Controlled



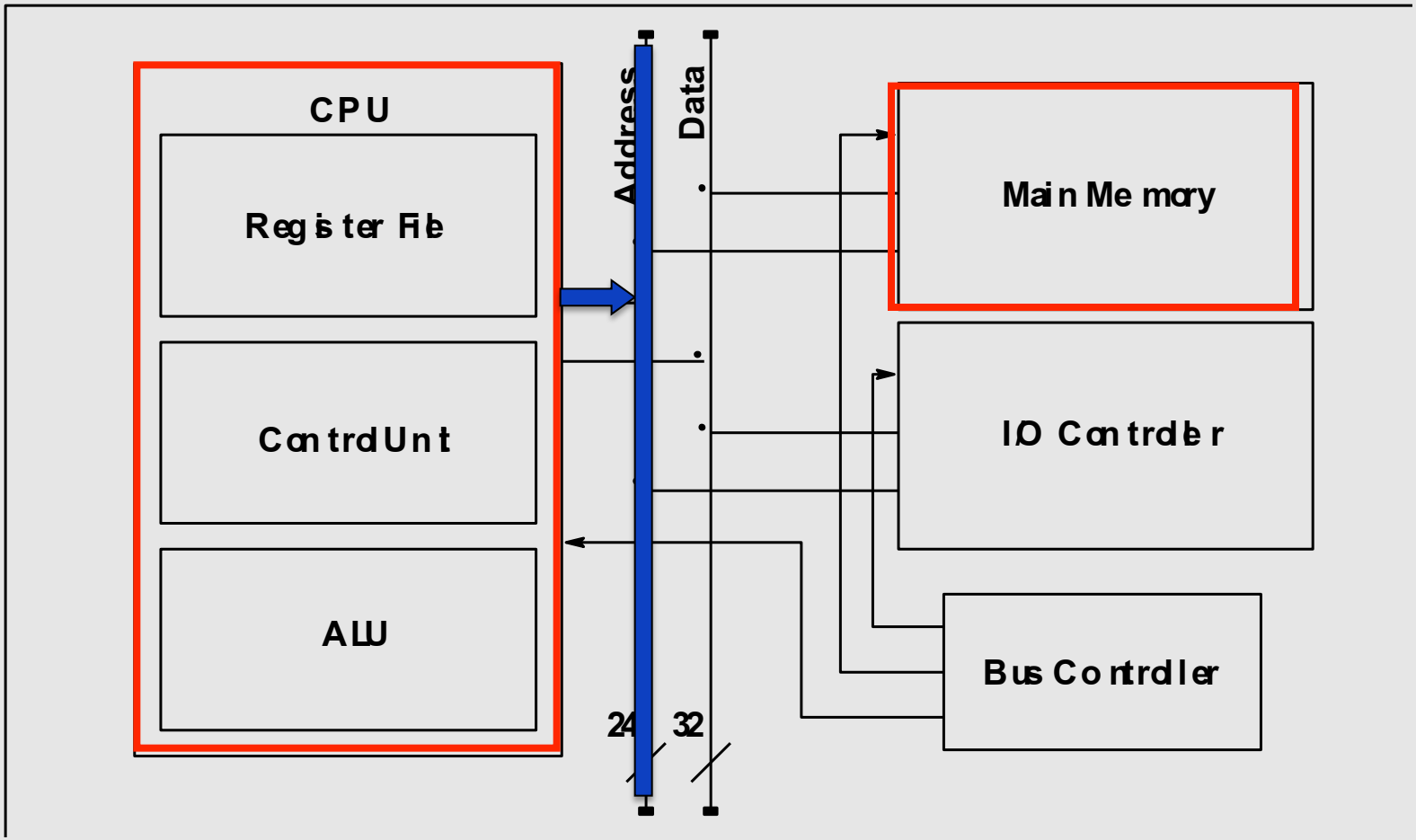
Centrally Controlled(CPU-Memory)



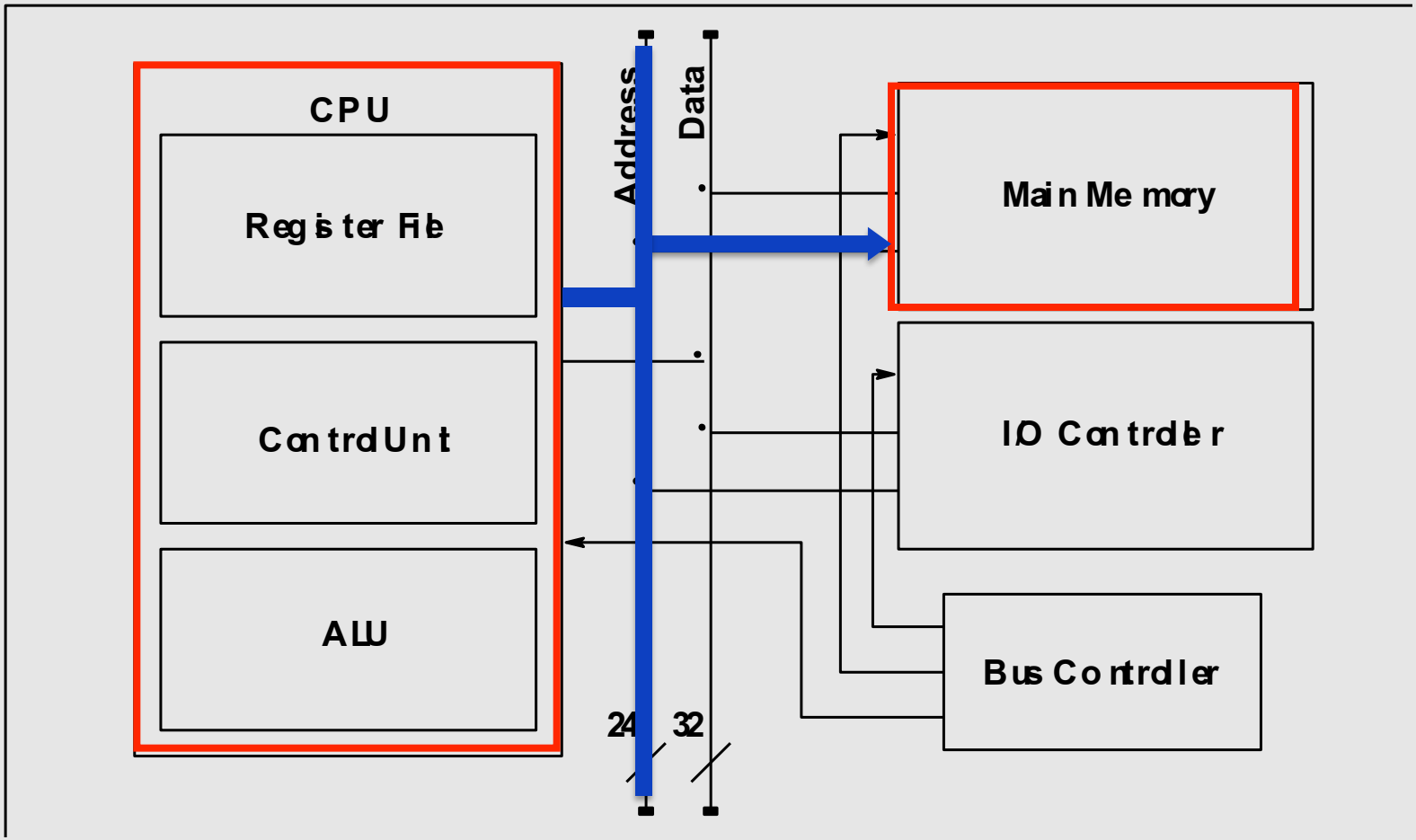
Centrally Controlled(CPU-Memory)



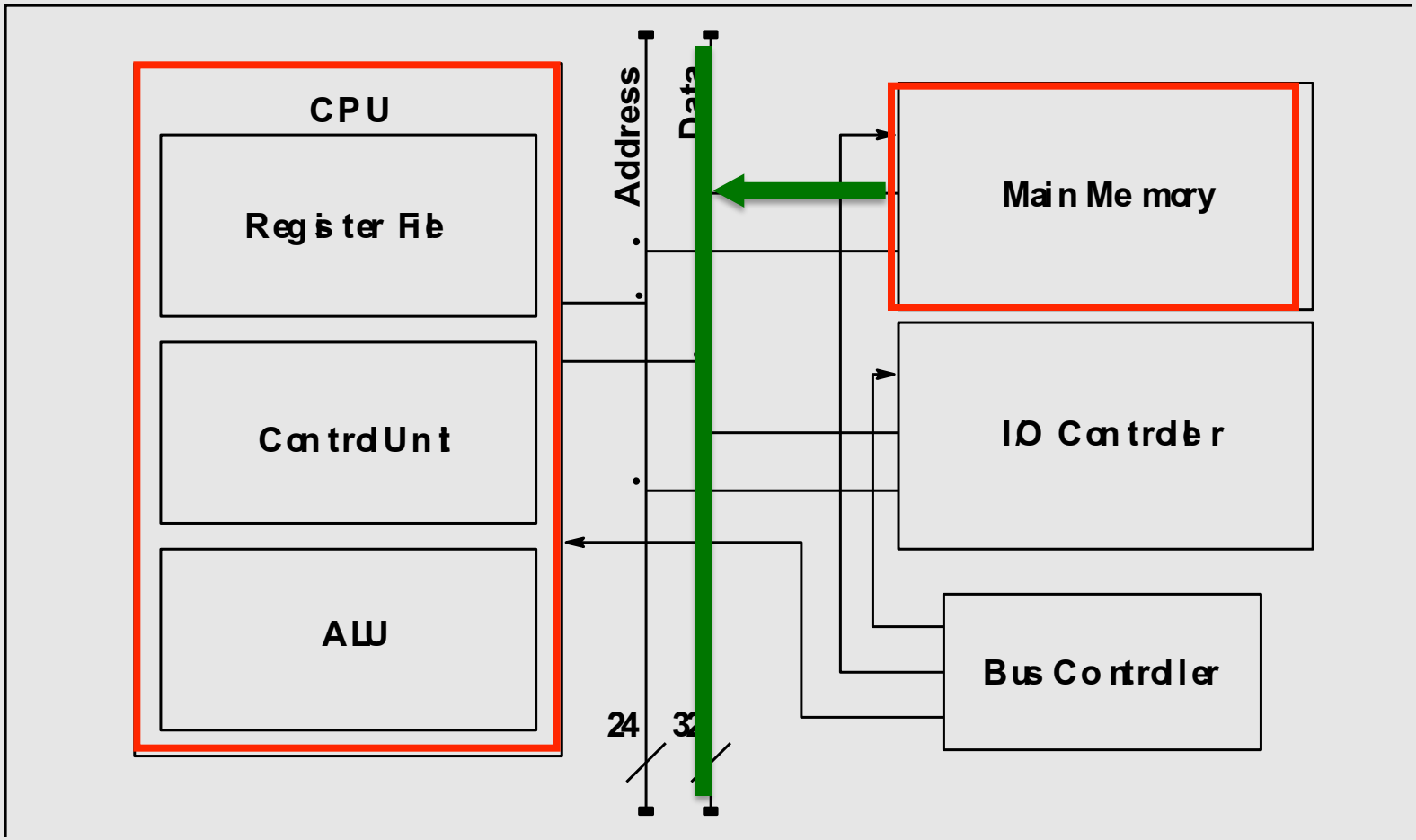
Centrally Controlled(CPU-Memory)



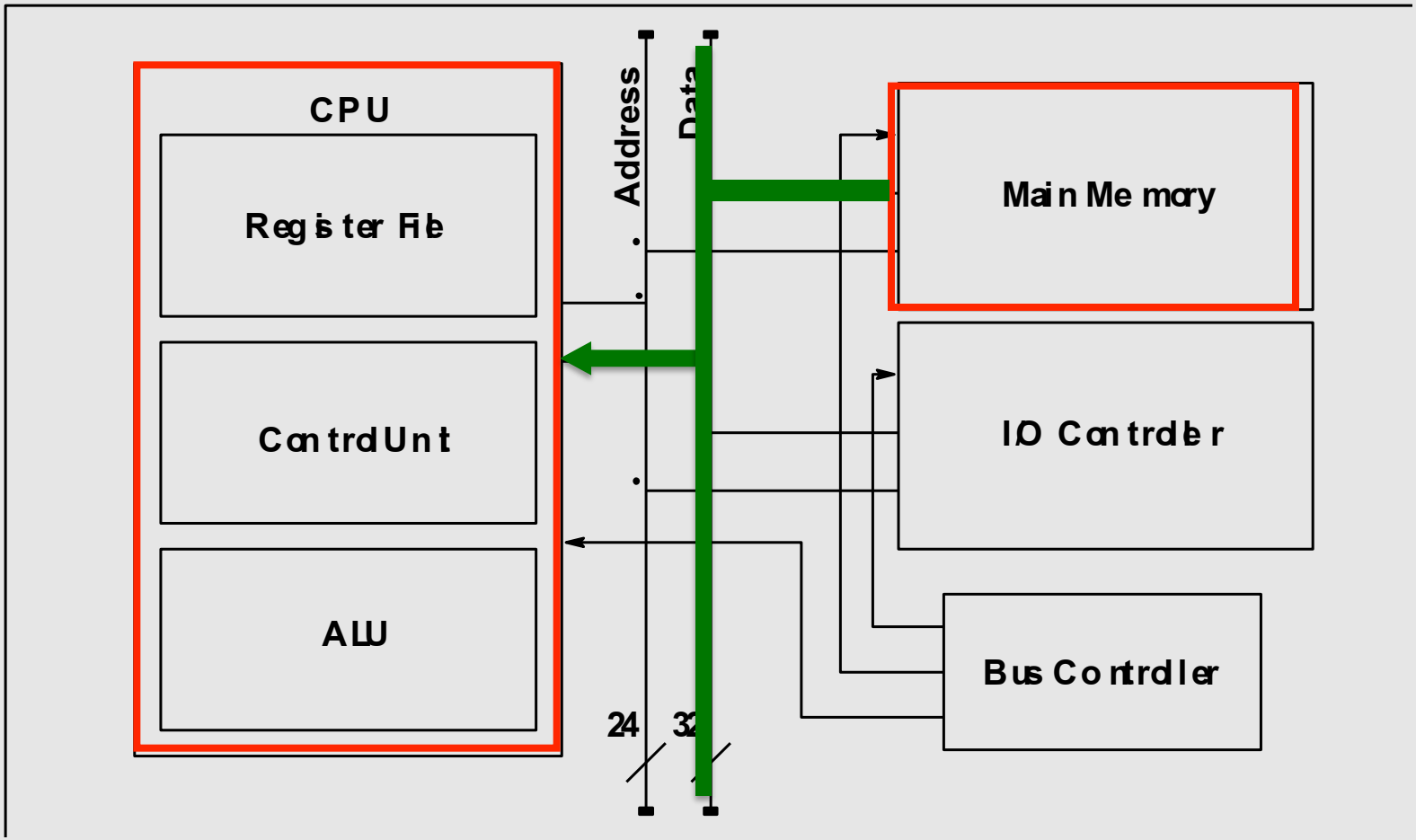
Centrally Controlled(CPU-Memory)



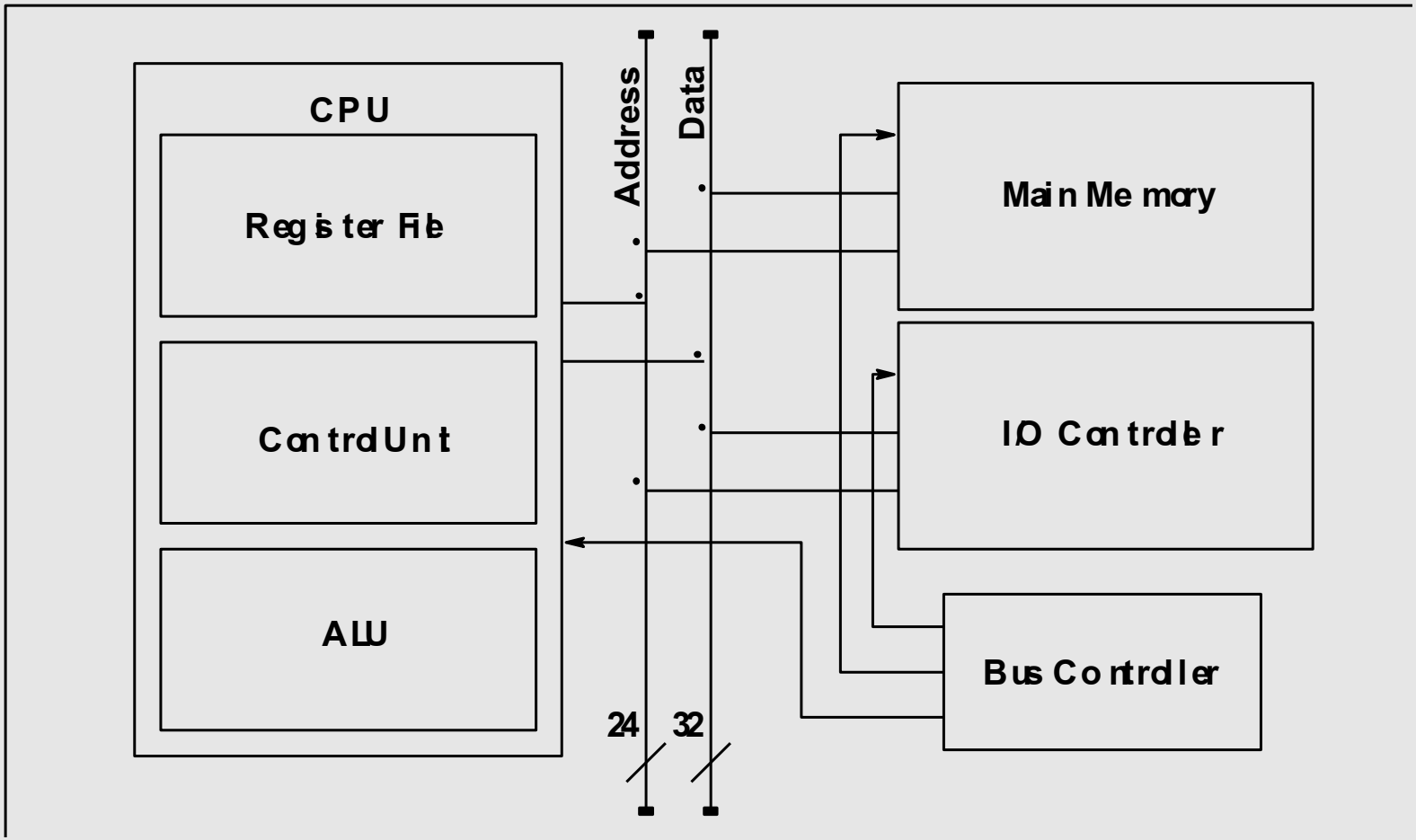
Centrally Controlled(CPU-Memory)



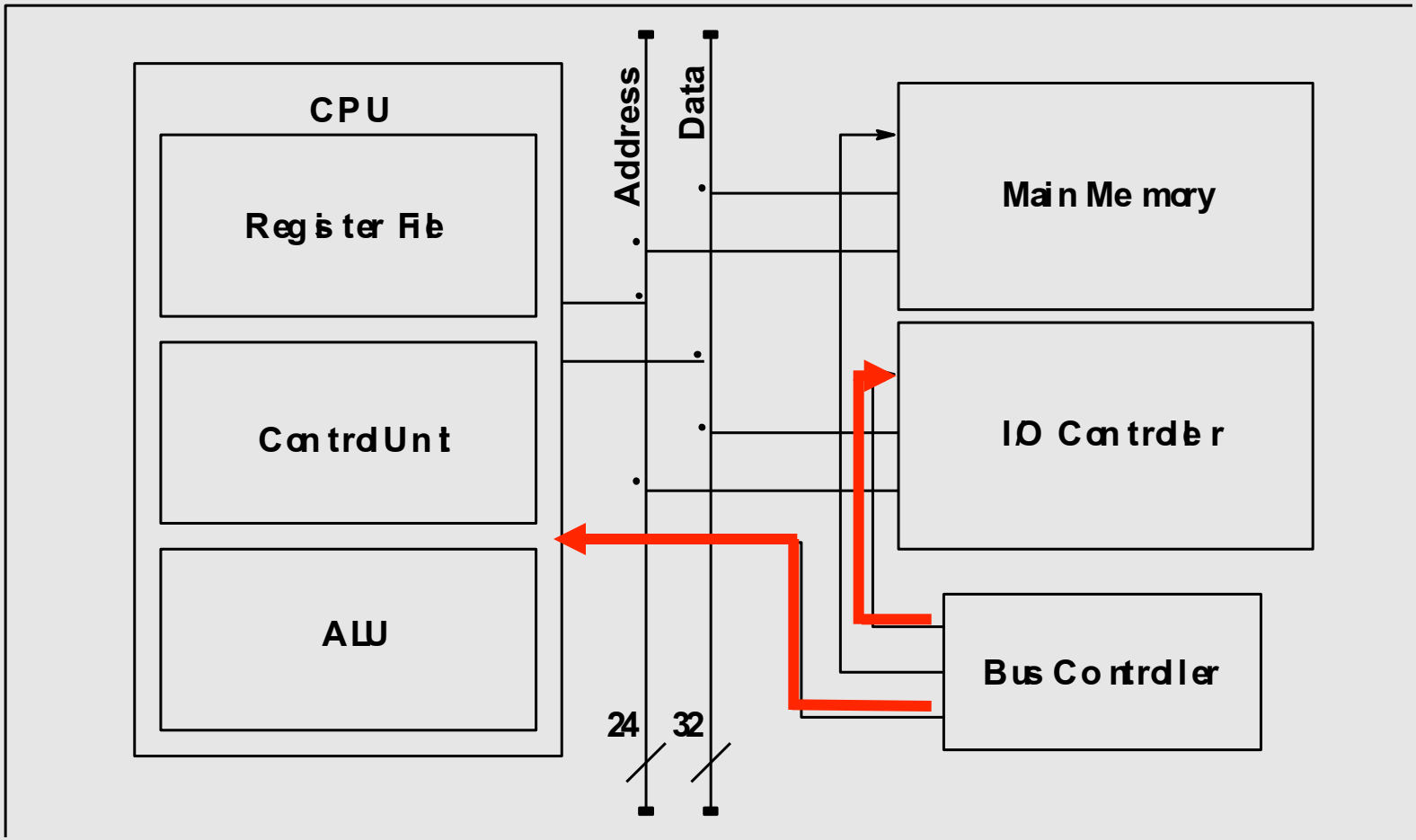
Centrally Controlled(CPU-Memory)



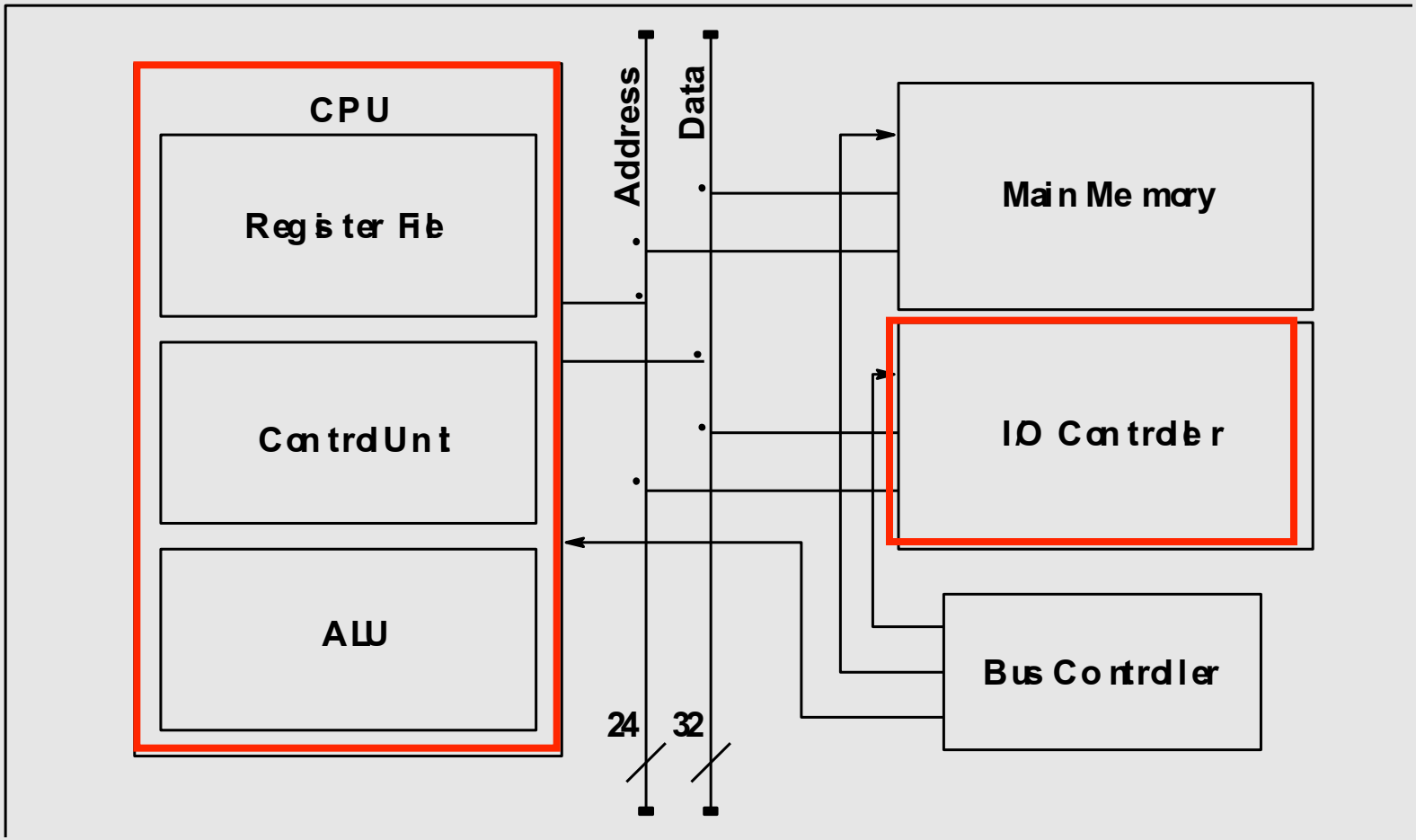
Centrally Controlled(CPU-I/O Controller)



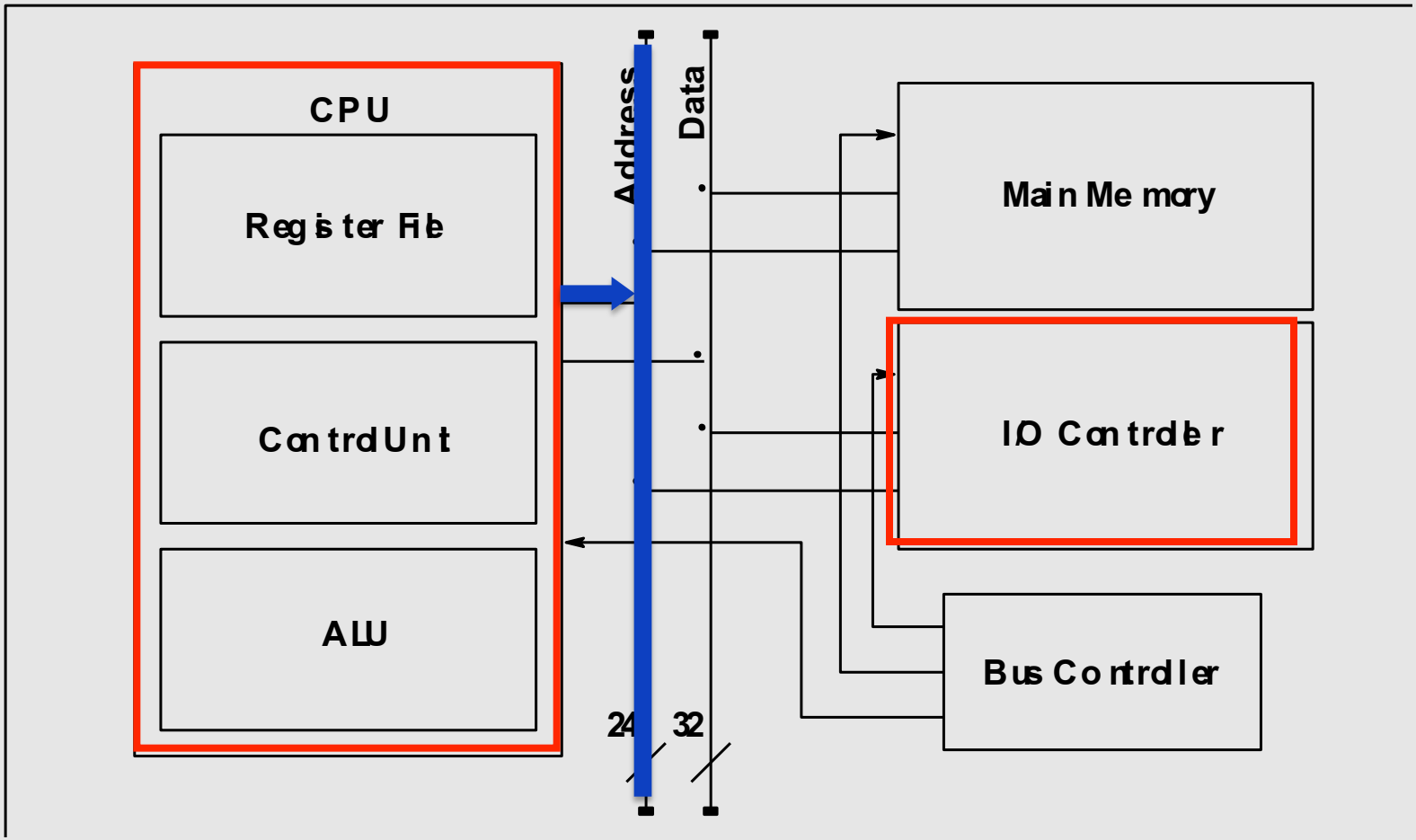
Centrally Controlled(CPU-I/O Controller)



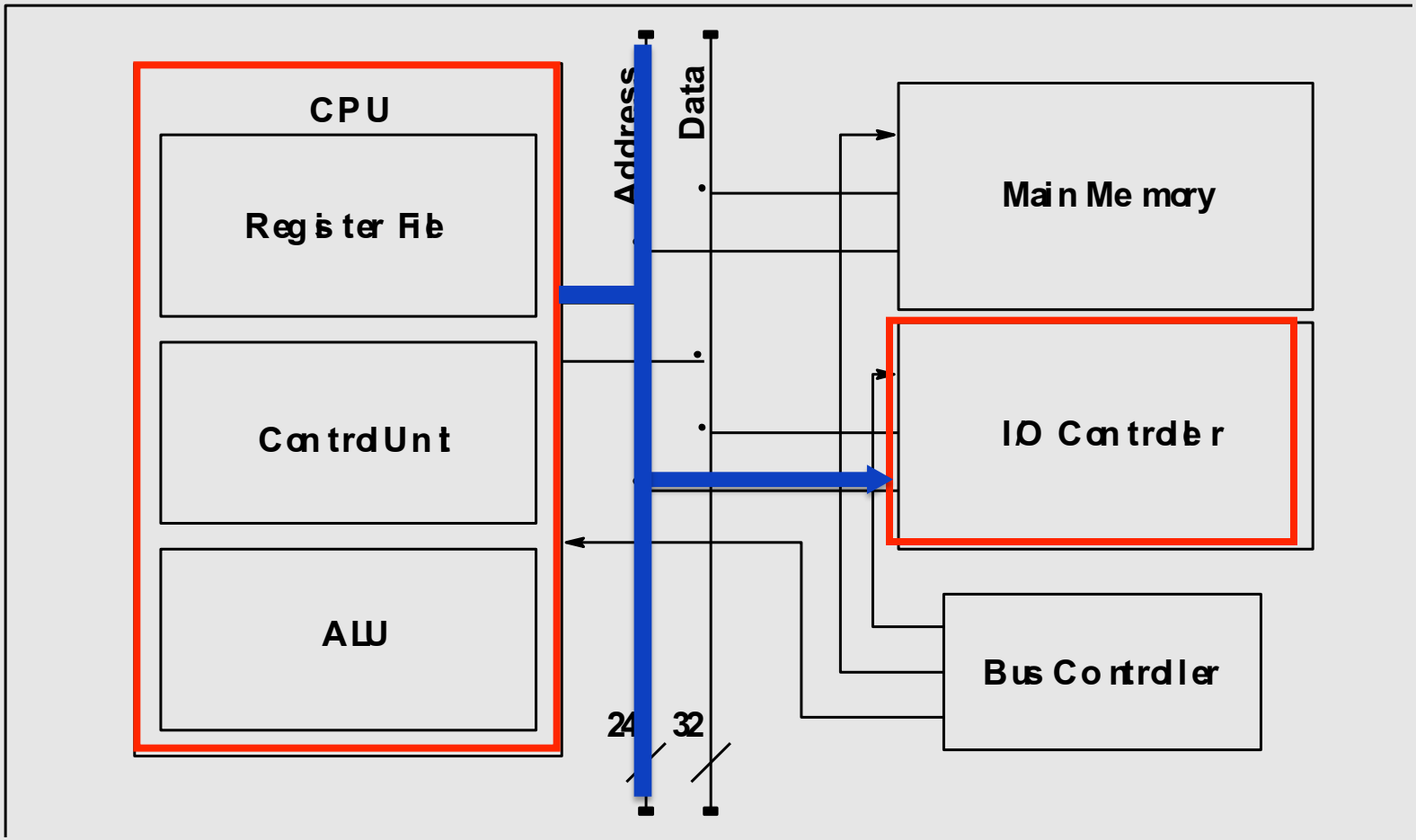
Centrally Controlled(CPU-I/O Controller)



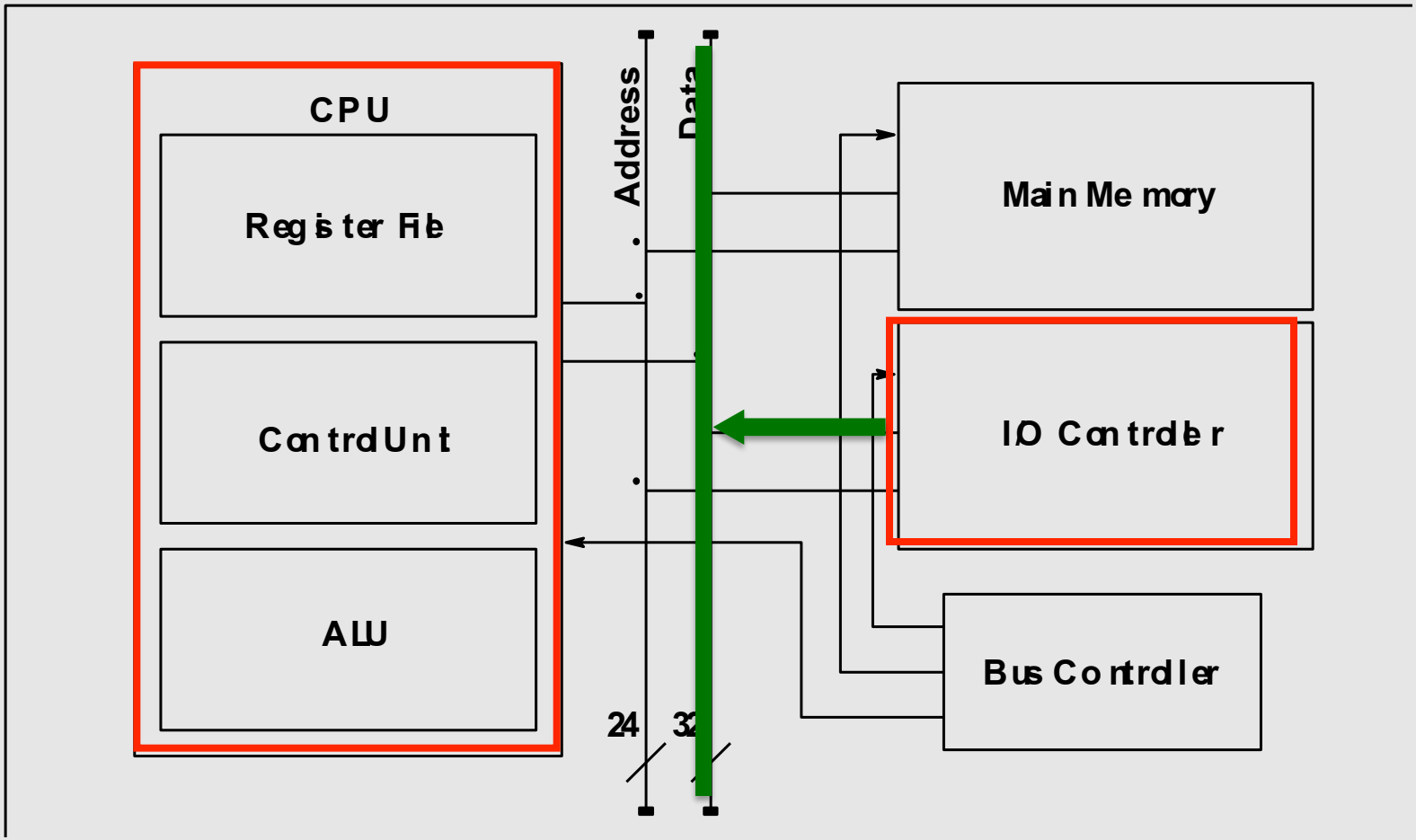
Centrally Controlled(CPU-I/O Controller)



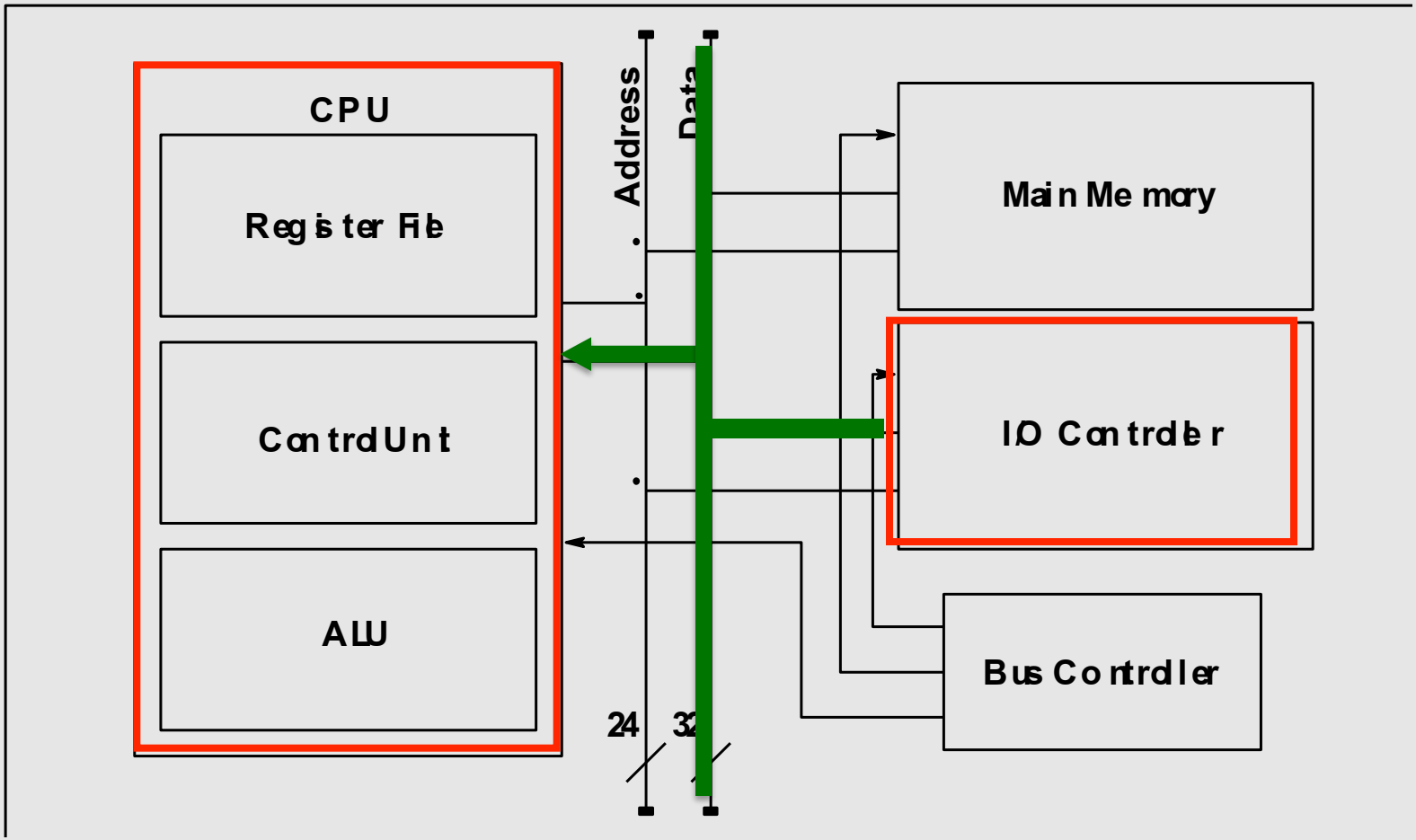
Centrally Controlled(I/O Controller-> CPU)



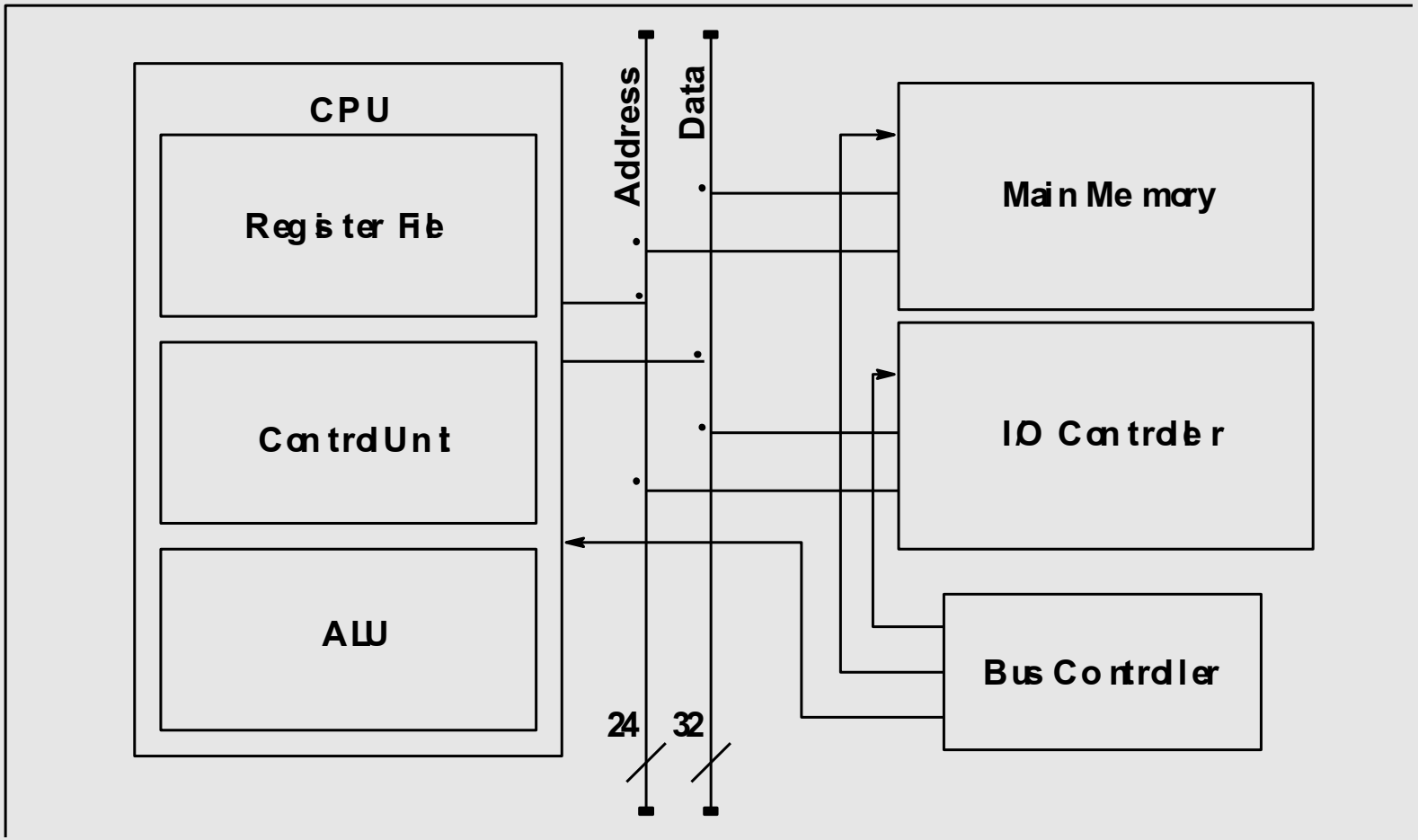
Centrally Controlled(CPU-I/O Controller)



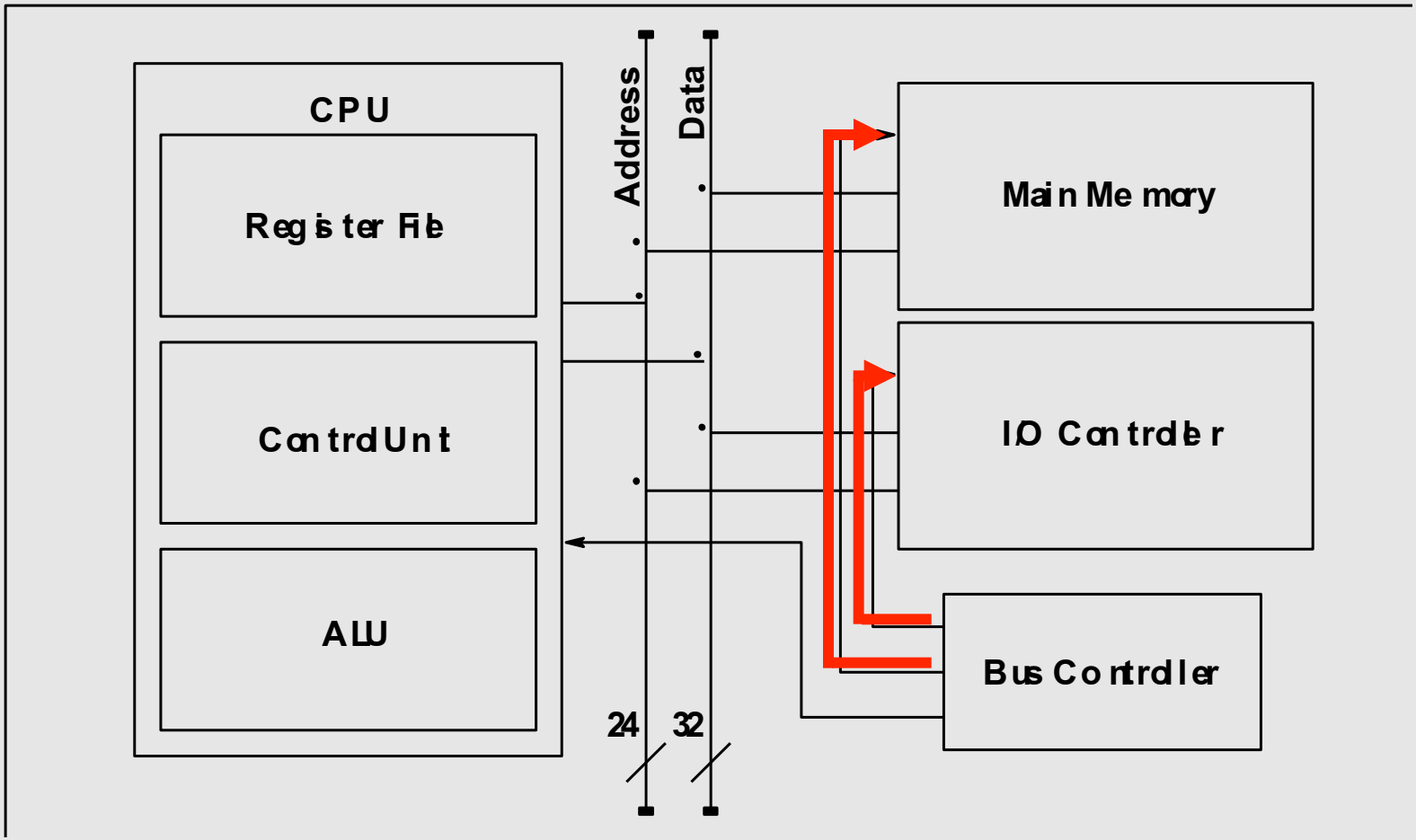
Centrally Controlled(CPU-I/O Controller)



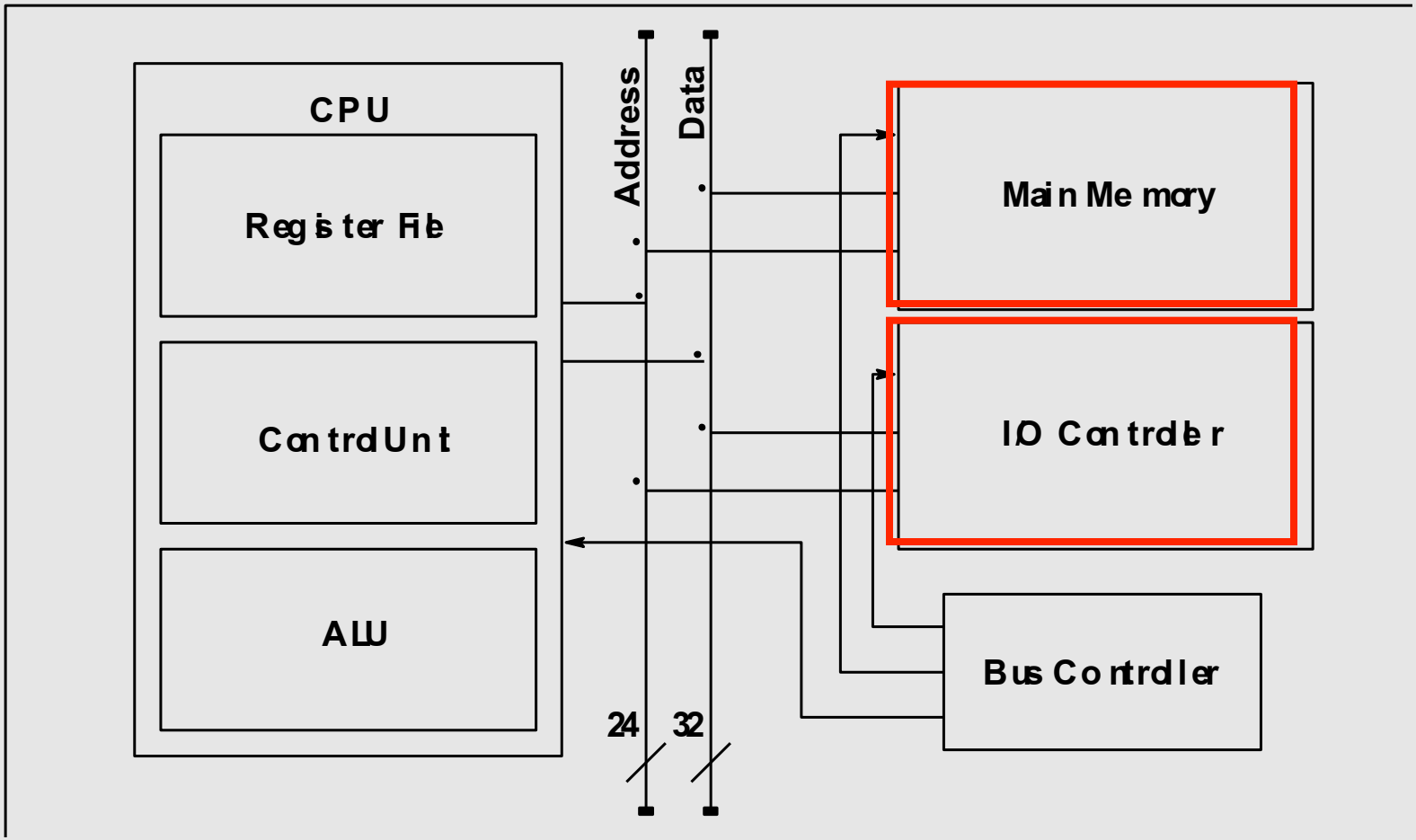
Centrally Controlled(I/O Controller-Memory)



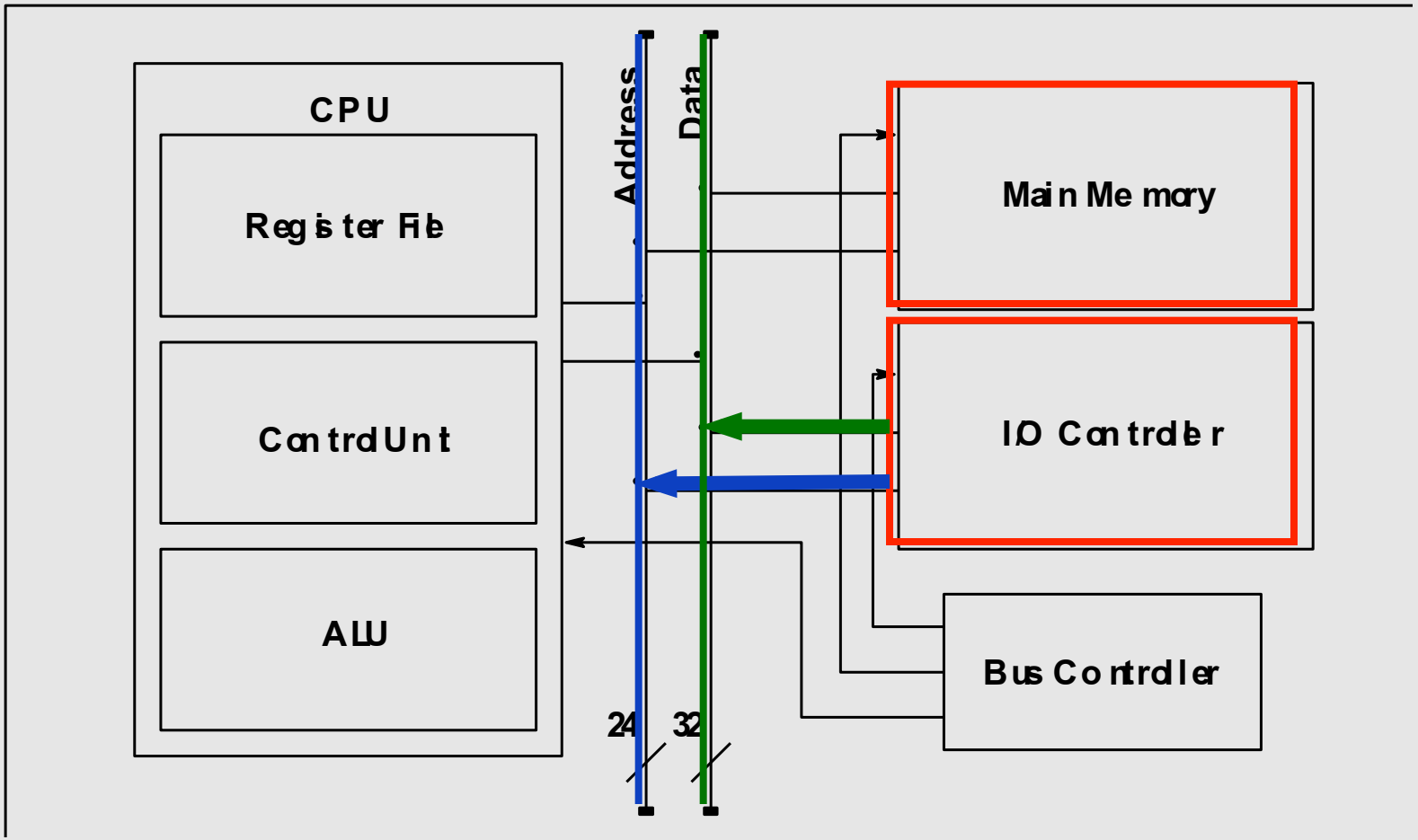
Centrally Controlled(I/O Controller-Memory)



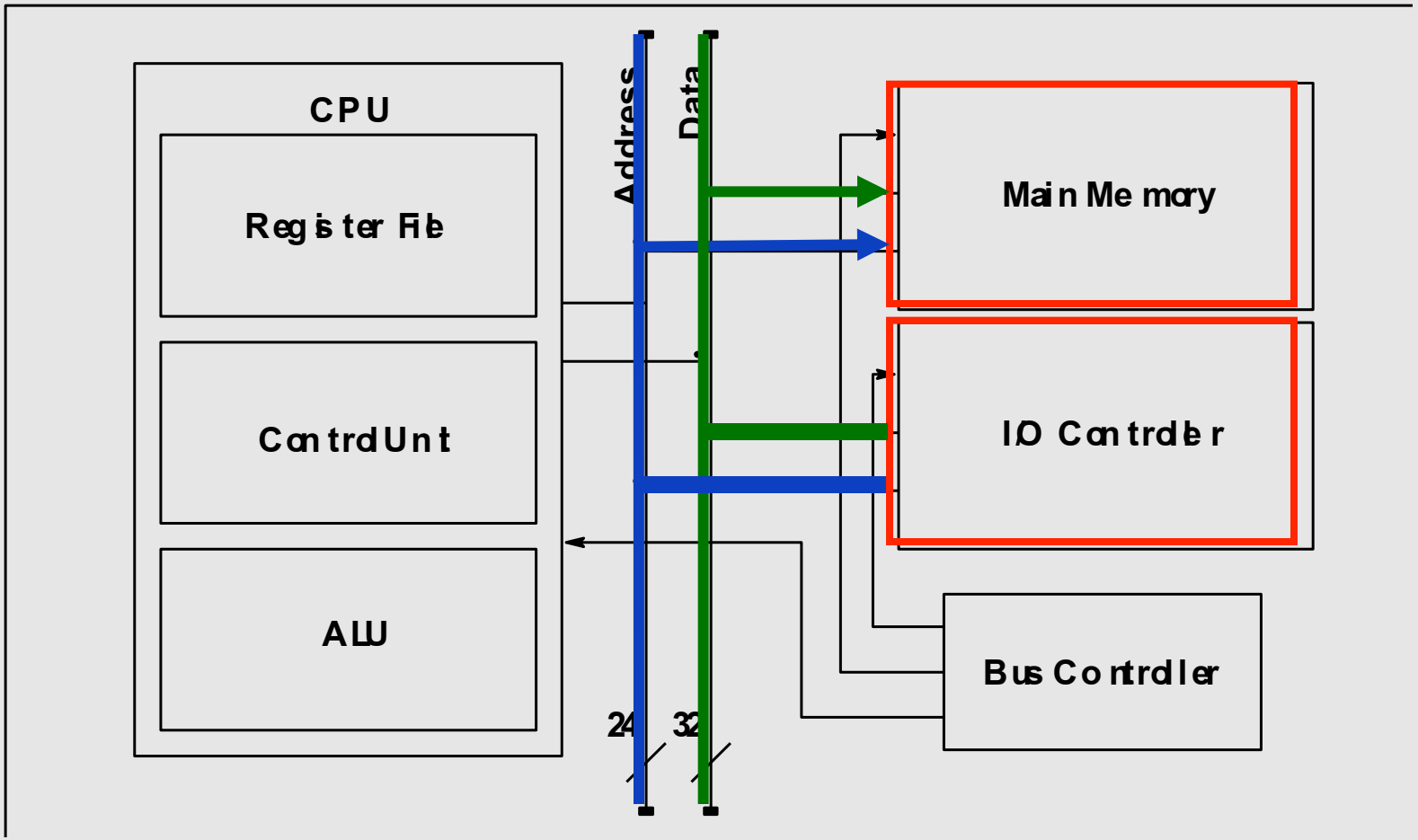
Centrally Controlled(I/O Controller->Memory)



Centrally Controlled(I/O Controller->Memory)



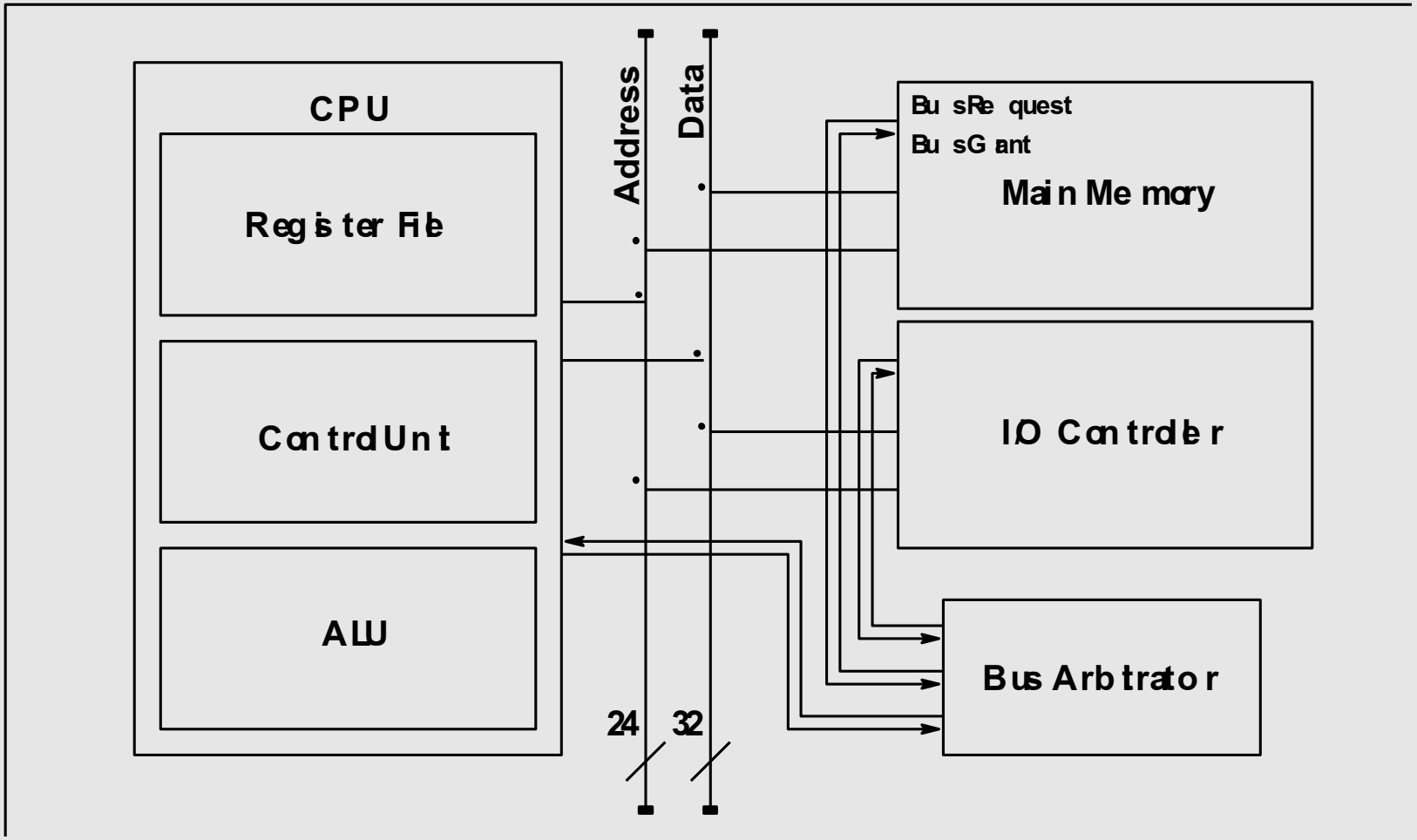
Centrally Controlled(I/O Controller->Memory)



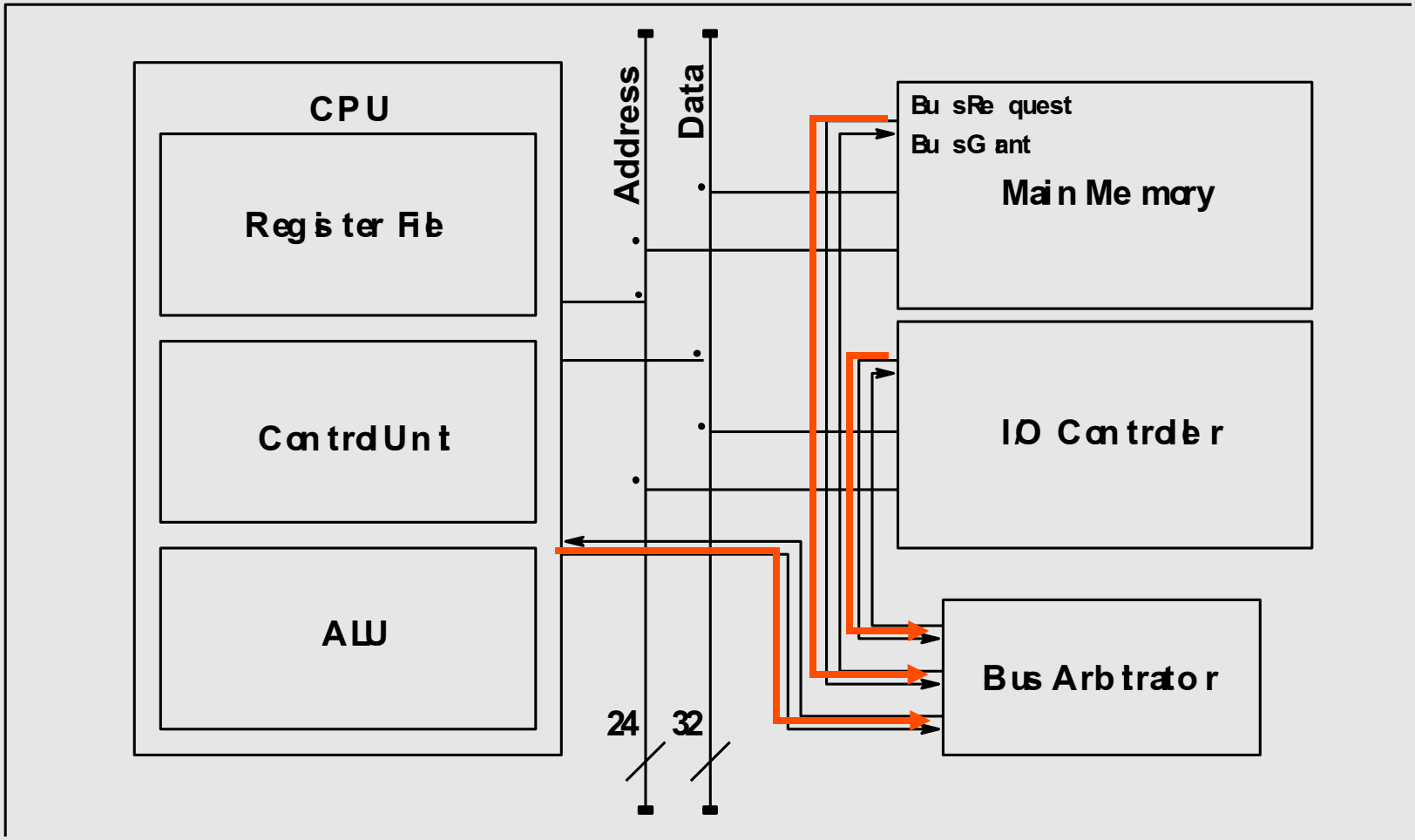
Bussing - Centrally Arbitrated

- In a centrally arbitrated bus, devices “arbitrate” for control of the bus, each becoming a “bus master” temporarily, to control reads or writes to other bus devices, which temporarily act as “bus slaves”.
- Arbitration is performed by a central “bus arbitrator”, which monitors “bus request” signals from each device on the bus.
- Following a rule defined by the bus protocol, the controller “grants” the bus to one of several requesting devices.

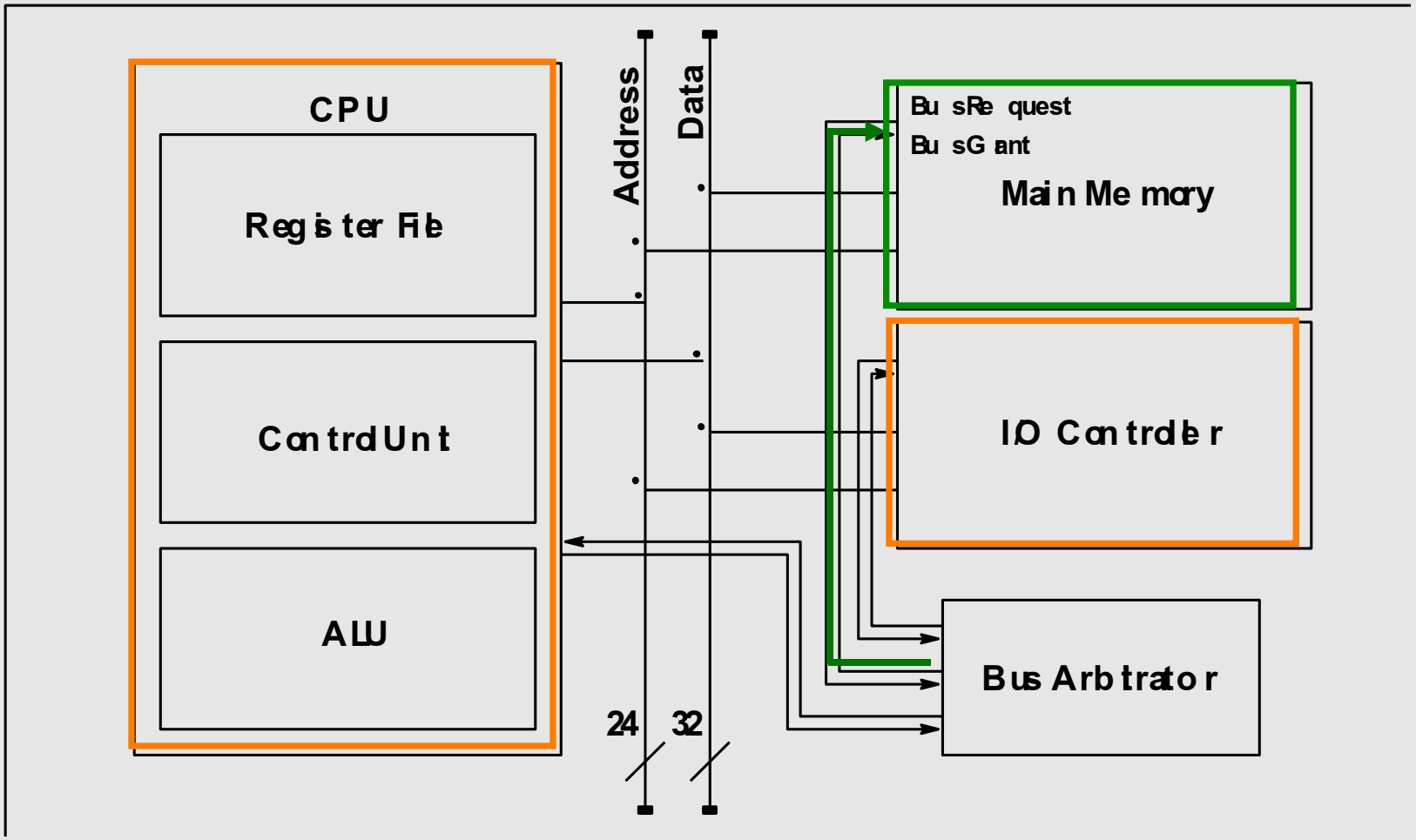
Centrally Arbitrated



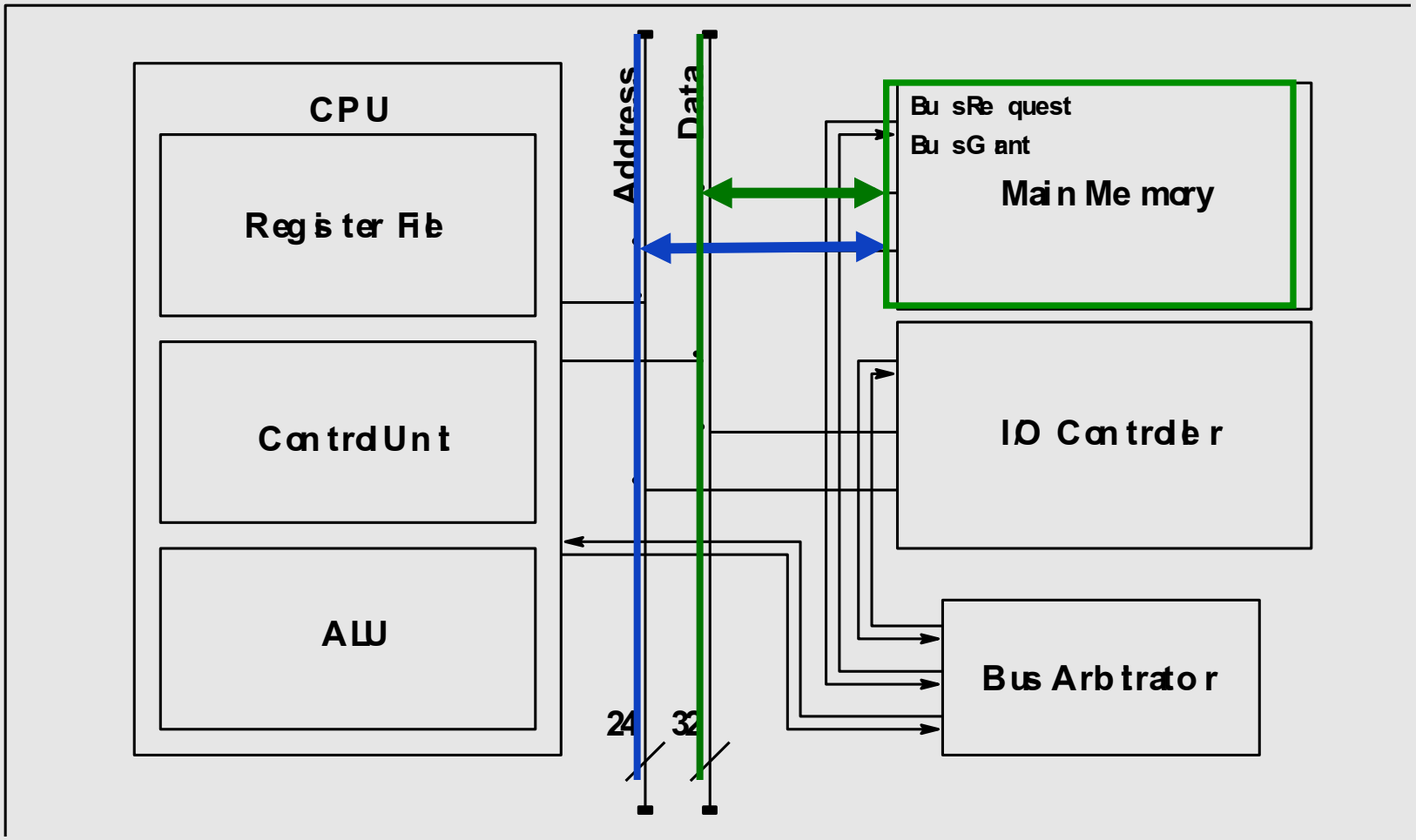
Centrally Arbitrated



Centrally Arbitrated



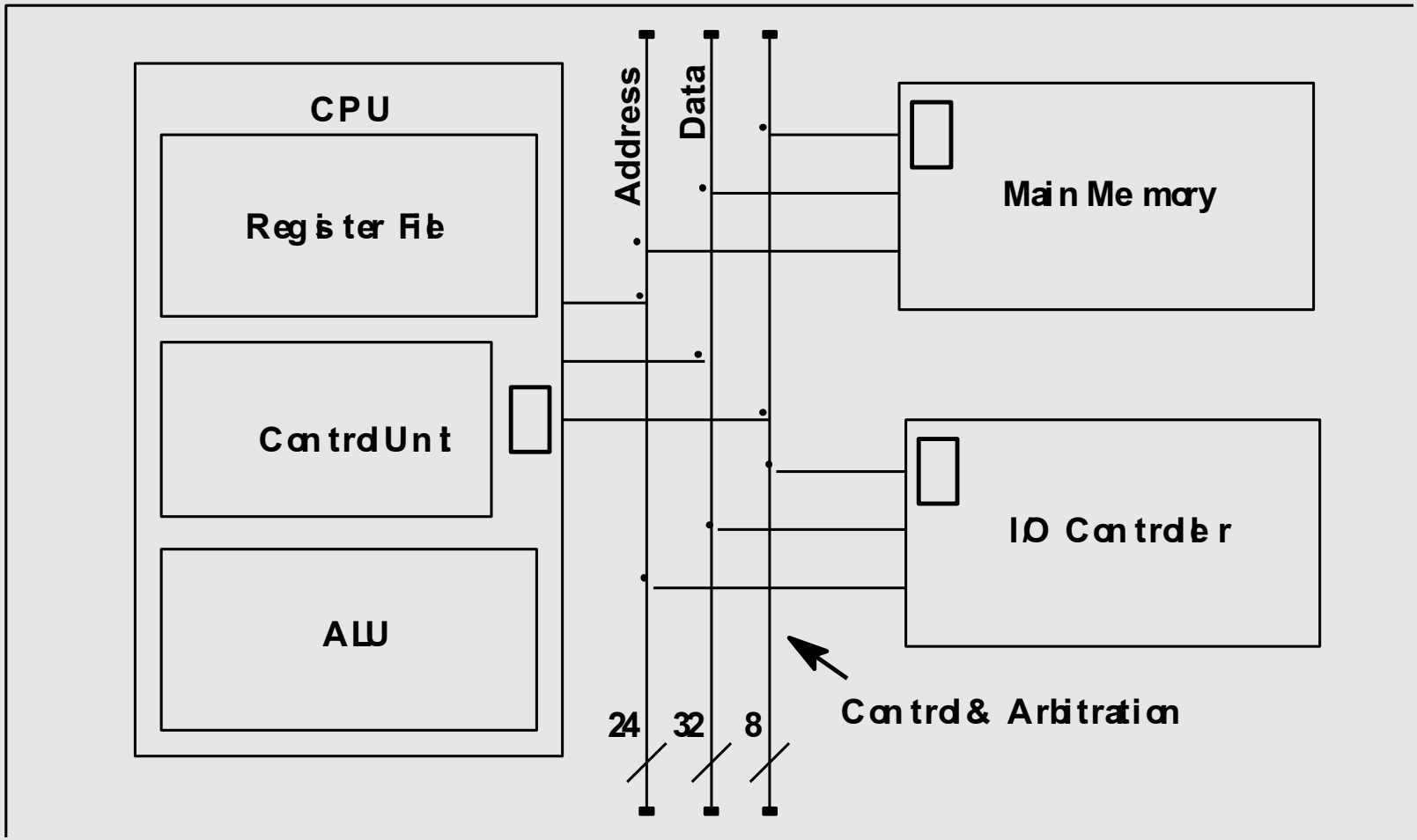
Centrally Arbitrated



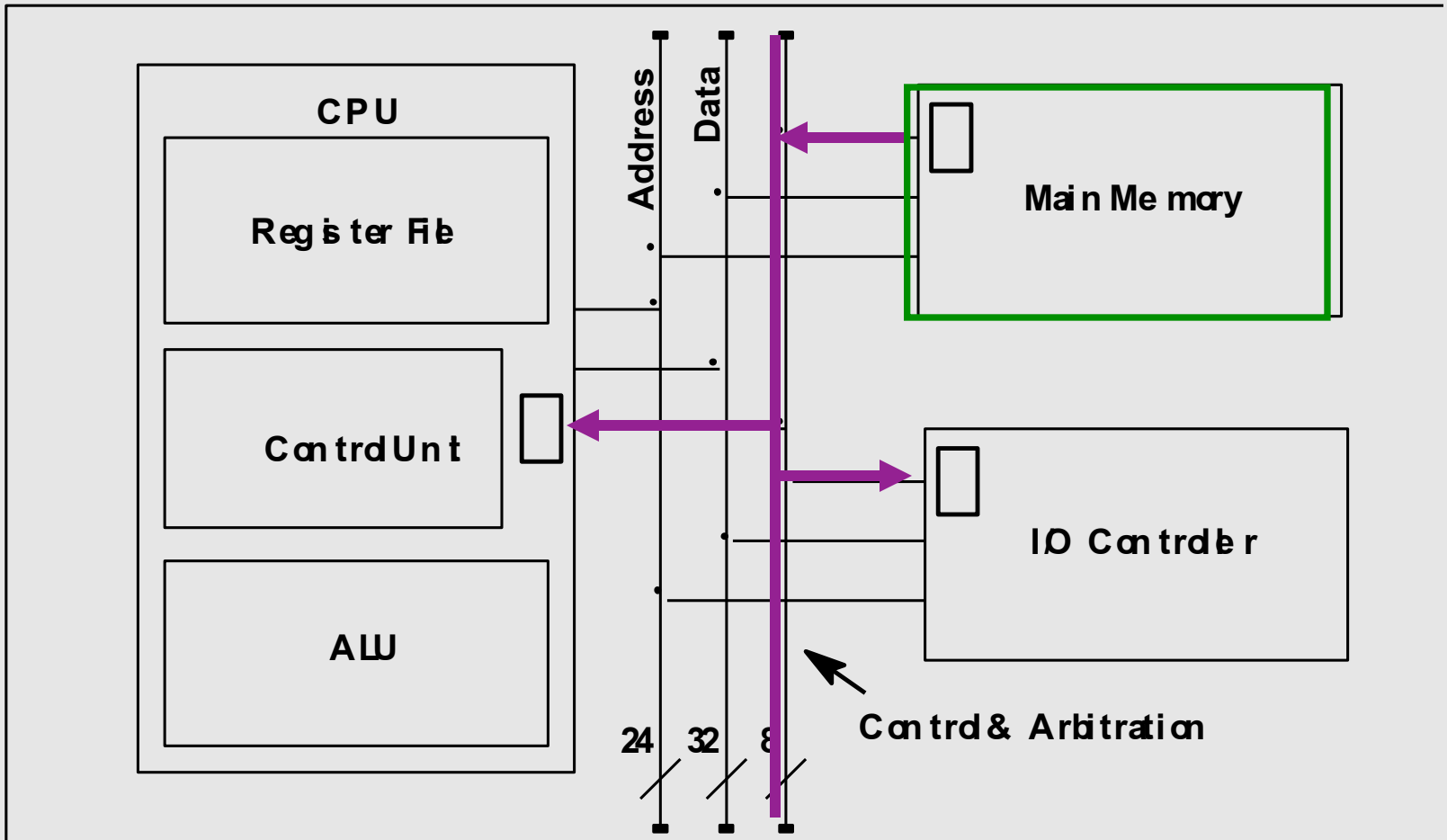
Bussing - Distributed Arbitration

- In a distributed arbitration bus, every bus device is “intelligent” and is capable of negotiating access to the bus itself.
- In such a bus, devices all broadcast to their peers their intent to use the bus, and following a rule defined by the bus protocol, devices accede to those who are to be given access.
- Busses in modern multi-processor systems are typically built in this manner.

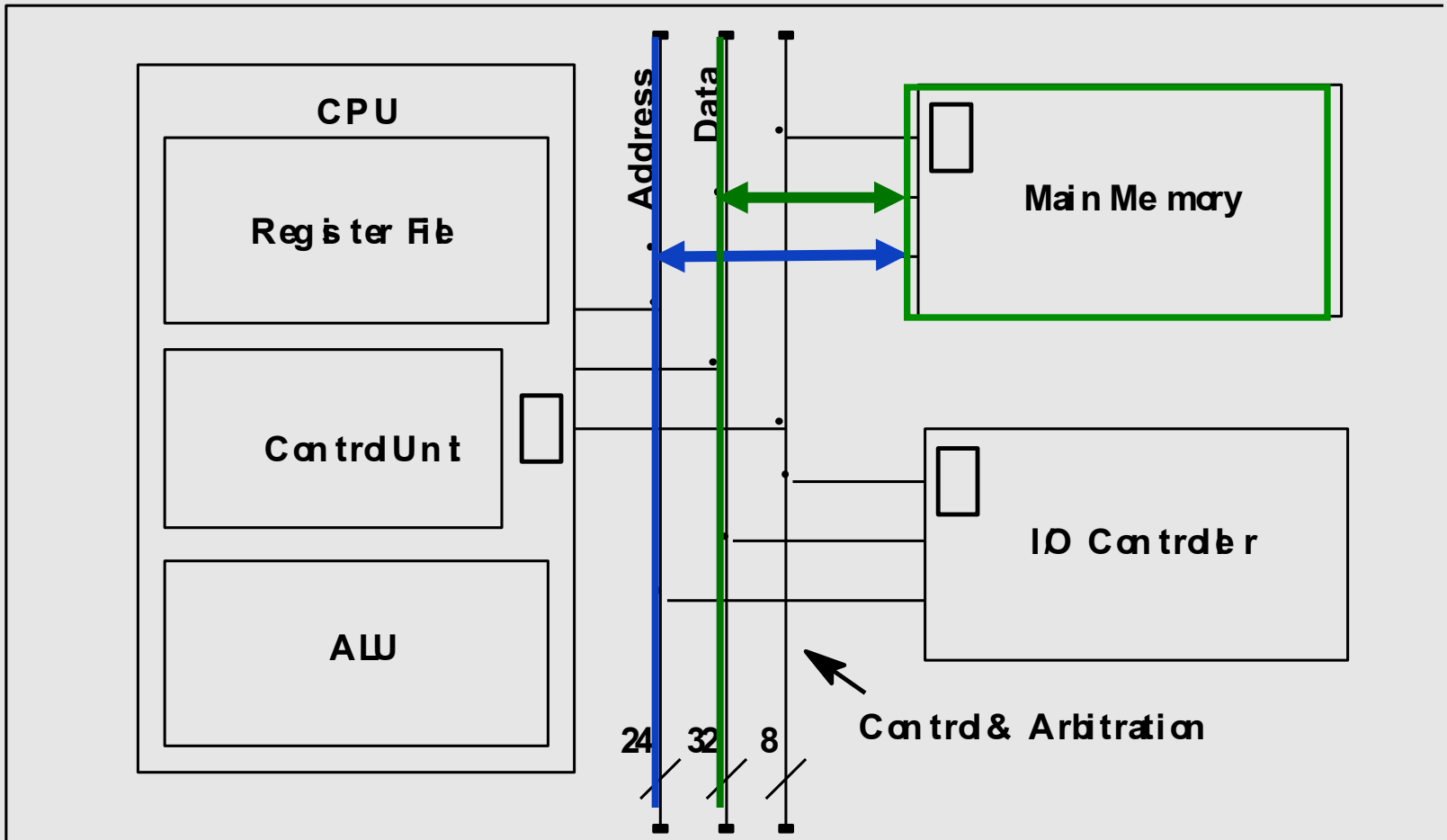
Distributed Arbitration



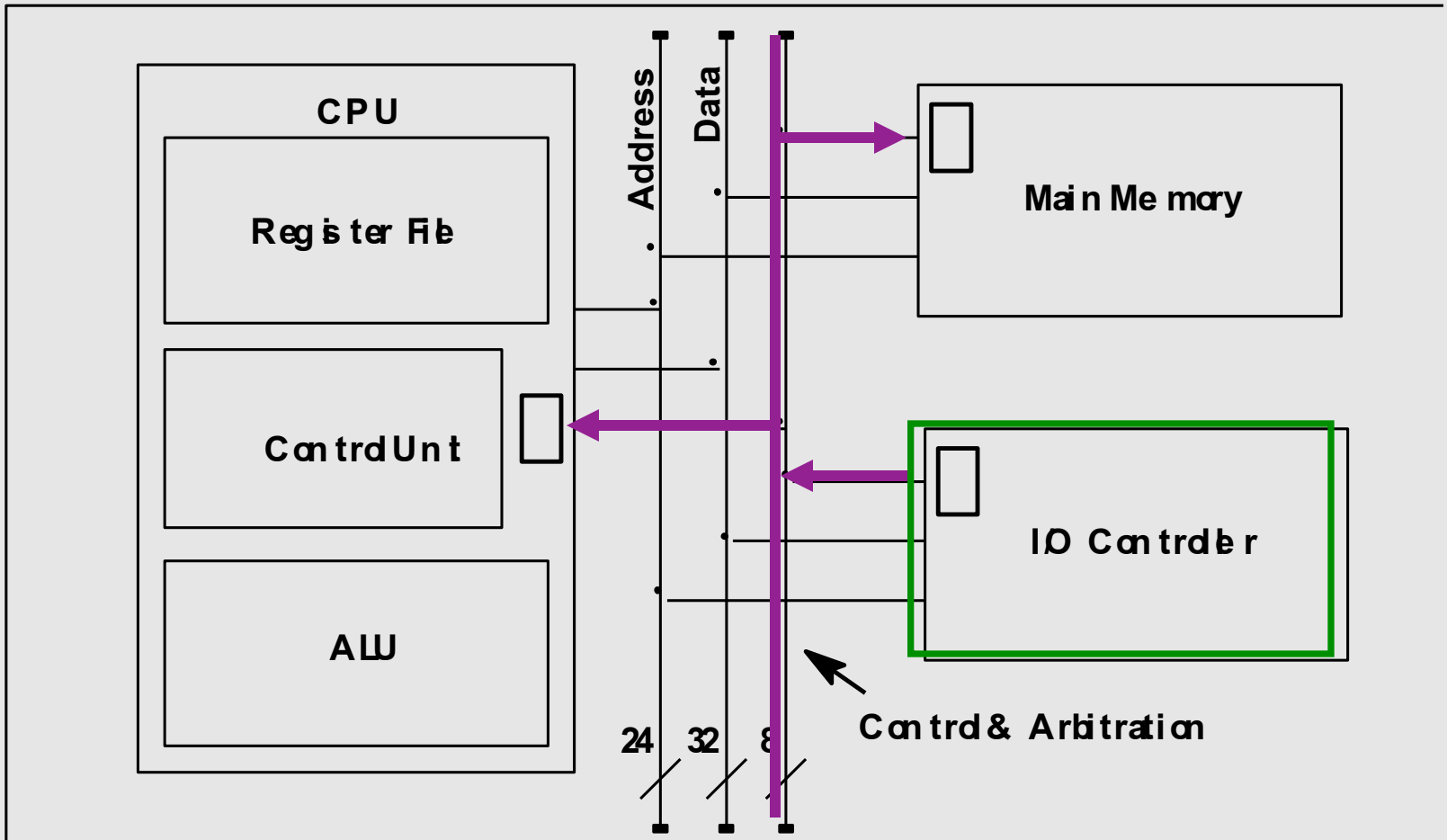
Distributed Arbitration (Memory)



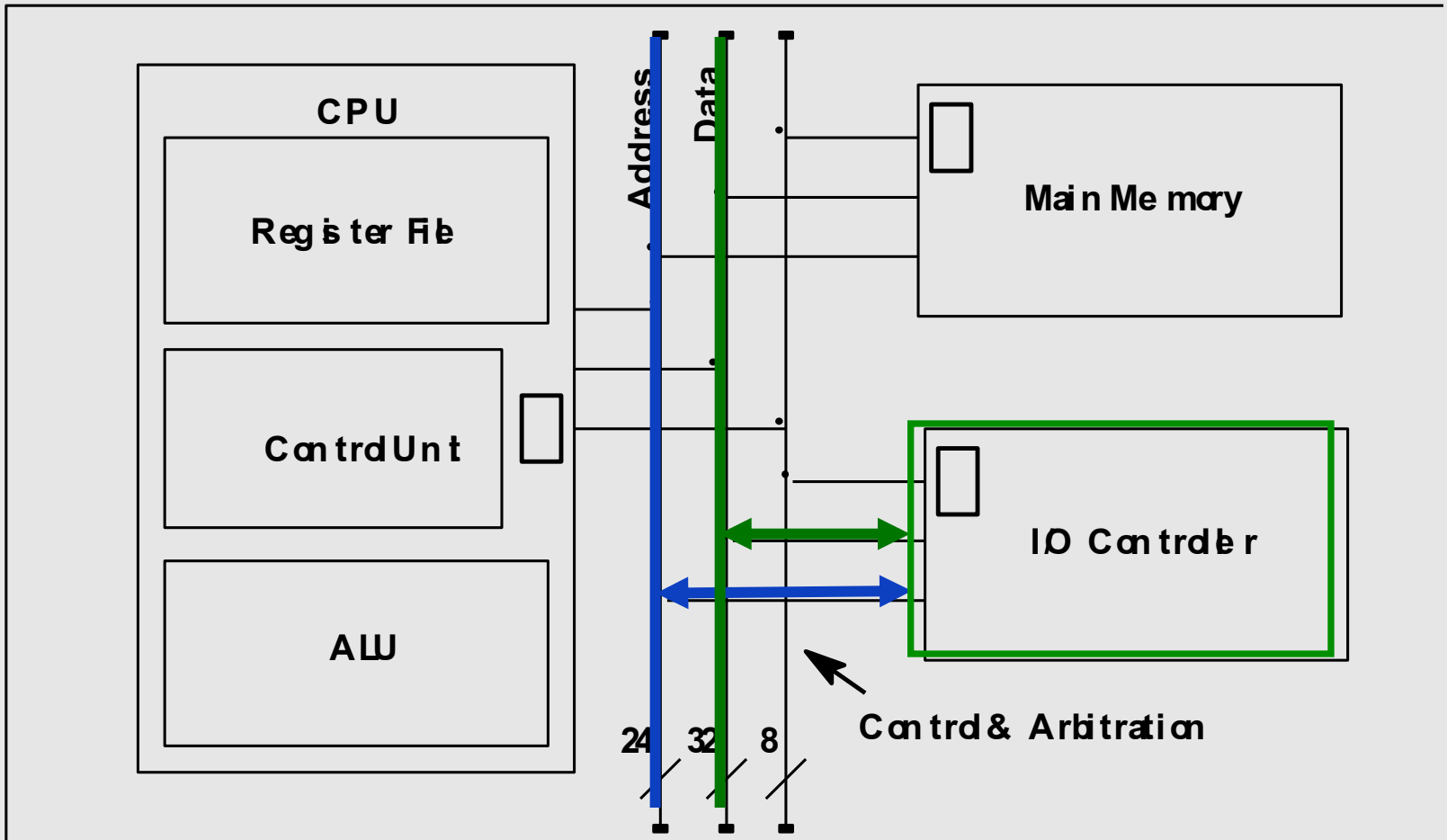
Distributed Arbitration (Memory)



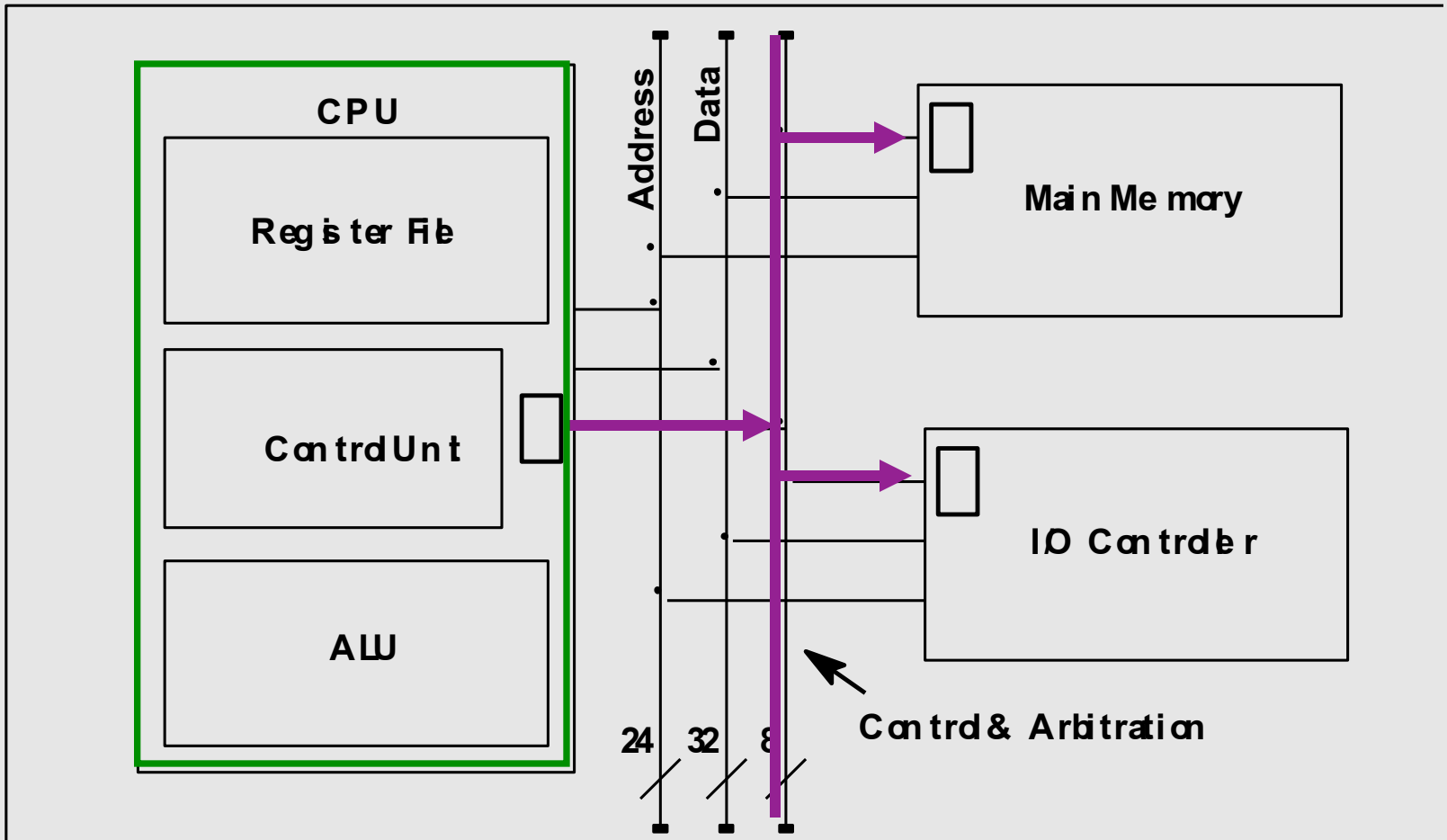
Distributed Arbitration (I/O Controller)



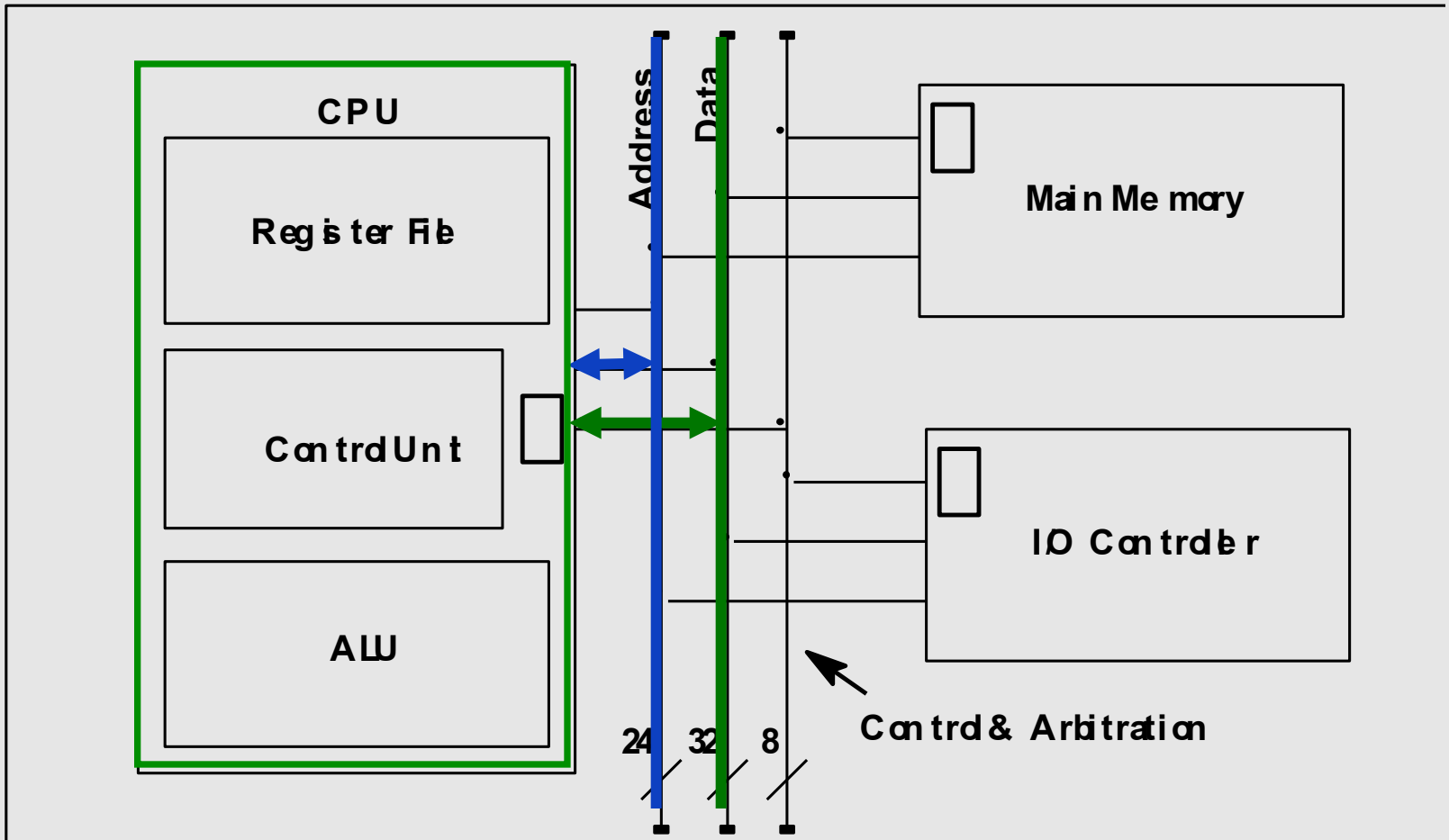
Distributed Arbitration (I/O Controller)



Distributed Arbitration (CPU)



Distributed Arbitration (CPU)



Bussing - Speed

- **The speed at which we can transfer data across a bus depends on several factors:**
  - *Bus Width* - how many data bits can we transfer at once ? Examples: 8, 16, 32, 64, 128 or 256 bits (data wires).
  - *Bus Cycle Time* - how long does it take to negotiate for the bus and transfer data ? Examples: microseconds to tens of nanoseconds.
  - *Bus Burst Size* - many busses will negotiate once, and then transfer blocks of data, termed “bursts”. The larger these are, the lower the average overhead per byte transferred.
  - *Bus Length* - in very fast busses, the length of the bus becomes a limiting factor to bus speed.

Some Bus Standards [Not Examinable]

- ISA Bus - devised for Personal Computers, an obsolete low performance bus now only used for "legacy" I/O devices.
- PCI Bus - standard I/O bus on IBM compatible Personal Computers, or other systems.
- PCI Express / PCI-E – current derivative of original PCI bus.
- AGP Bus – obsoleted but still used for graphics adaptors.
- VME Bus - obsolete I/O bus, but still the most common bus in industrial and embedded computers.
- SCSI Bus – still used for large peripherals such as disk arrays, tape drives, optical jukeboxes.
- USB and Firewire – Serial "busses" for peripherals.
- FSB or "Front Side Bus" – generic term for CPU to memory and graphics hardware.

Bus Standards Cont ...

- **Each of these standards defines:**
  - Number of data lines.
  - Number of address lines.
  - Number and type of control lines.
  - Timing behaviour of signals.
  - Bus protocol behaviour.
  - Electrical characteristics of bus drivers and receivers.
  - Connector types and pinouts for bus interface.

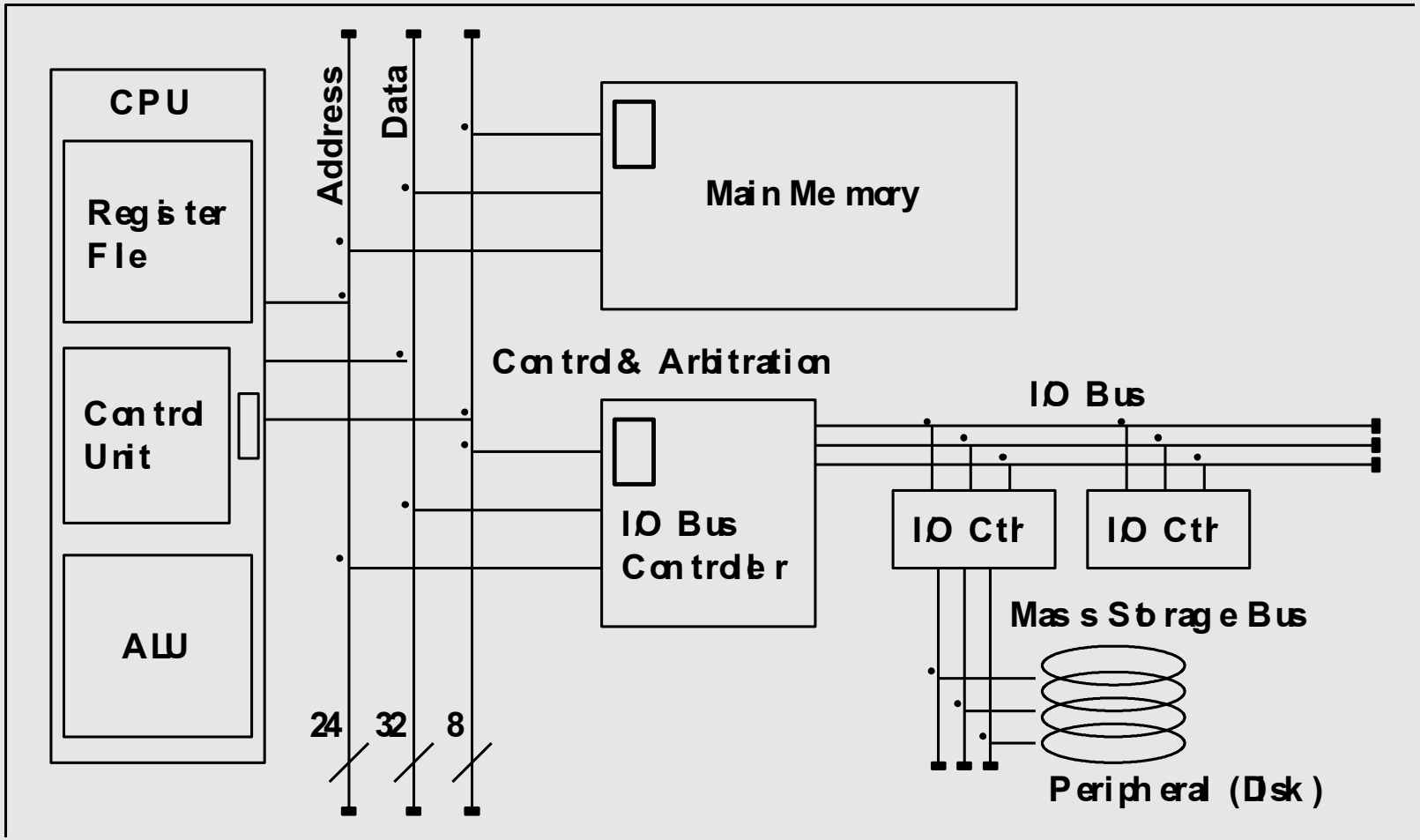
Bus Hierarchies

- Most computers use multiple busses, each for specialised purposes.
- A CPU typically has one or more *internal busses* to connect the ALU and register file(s). Such busses are controlled by the CPU control unit.
- A main *System Bus* is usually employed to connect a CPU with the Main Memory, and an I/O Bus Controller.
- The "Front Side Bus" in many PC motherboards is an example of a *System Bus* designed to connect the processor to main memory and Graphics processors.
- In modern designs a *System Bus* has a wider "datapath" than an I/O bus, so it can transfer more data per bus operation, and thus transfer data much faster.

Bus Hierarchies

- The I/O Bus Controller will usually translate between the System Bus protocol, and an I/O Bus protocol.
- The I/O Bus has multiple I/O controller devices connected to it (eg PCI-E boards in a PC).
- Mass Storage Controllers (Disk/Tape) then usually drive an I/O bus such as USB, Firewire, or SCSI to the peripherals storing the data.
- Typical machines have hierarchies of 2-4 busses.

Multiple Busses



Summary / What's next?

- **Basic organisation**
- **Concepts in fast addition and multiplication.**

- **Bussing Concepts**
- **Address, Data and Control busses.**

- **Control**
  - Microcoded & Hardwired Machines