Digital microsystems design

Marius Marcu

2020

Objectives

- Specific objectives
  - Internal architecture and external interfaces of x86 processors and their functional behavior

Outline

- Pins
- Machine cycles
- Data transfer
  - Normal cycles
  - Pipelined cycles
  - Burst cycles
- Pipeline
- System architecture
- Summary

80x86 microprocessors family

- 8086
  - Release year: 1978
  - First 16 bits microprocessor
- 80386
  - Release year: 1985
  - 32 bits microprocessor
- Pentium
  - 32 bits addresses/ 64 bits data
- Core
- x86 microprocessors family
  - IA-16
  - IA-32 (i386)
  - IA-32-64 (amd64)
- IA64 – Itanium
  - No IA32 compatible

8086

- Characteristics
  - 16 bits internal registers
  - 16 bits external data bus
  - 20 bits external address bus
    - Address space: 1 MB
    - 20 bits memory address space
    - 16 bits I/O address space
  - Multiplexed address and data lines
  - 5-10 MHz
  - 40 pins DIL package - Socket

80386

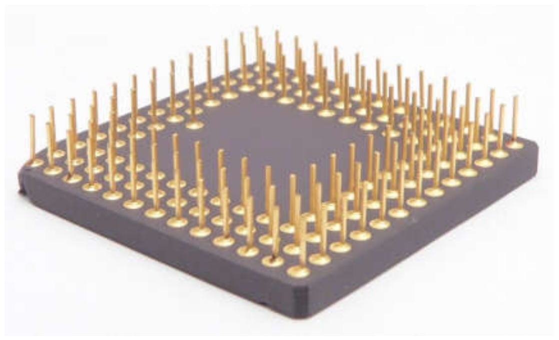
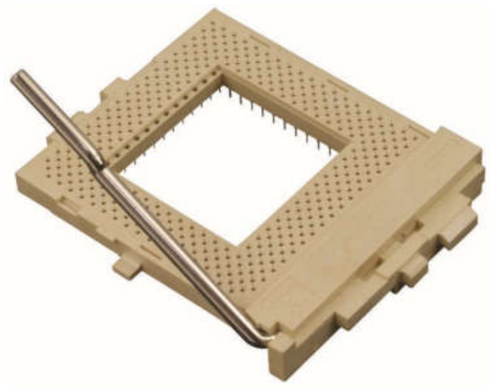
- Characteristics
  - 32 bits internal registers
  - 32 bits external data bus
  - 32 bits external address bus
    - Address space: 4 GB
    - 32 bits memory address space
    - 16 bits I/O address space
  - Non-multiplexed address and data lines
  - 12-33 MHz
  - 132 pins PGA package

Pentium

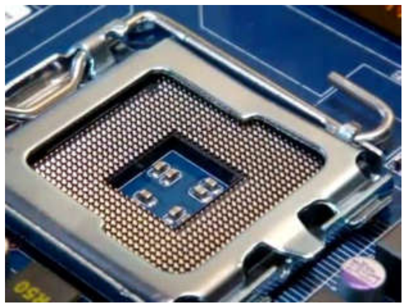
- Characteristics
  - 32 bits internal registers
  - 64 bits external data bus
  - 32 bits external address bus
    - Address space: 4 GB
    - 32 bits memory address space
    - 16 bits I/O address space
  - Non-multiplexed address and data lines
  - 60-66 MHz
  - 273 pins PGA package – Socket 4

Processor sockets

- PGA – pin grid array

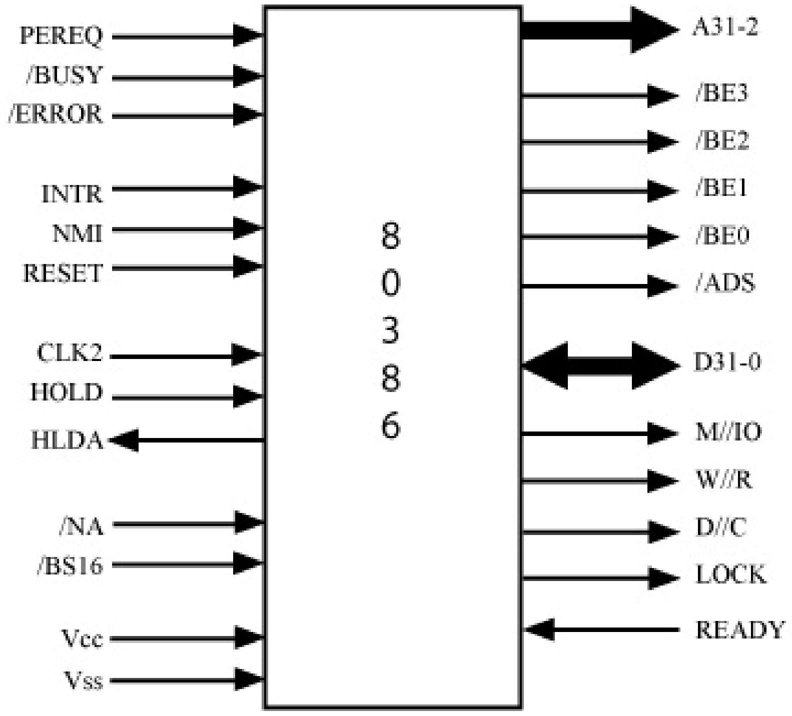



- LGA – land grid array



Processor pins

- 80386 local bus



Processor pins

- 80386

Name	Type	Description
A31-2	O, 3-state (Z)	Memory-I/O address bus
D31-0	I/O, Z	Data bus
/BE3-0	O, Z	Byte enables
/ADS	O, Z	Address status
D/ /C	O, Z	Data/Control indication
W/ /R	O, Z	Write-Read control indication - the processor is performing a memory or I/O write/read cycle
M/ /I/O	O, Z	Memory-I/O indication - distinguishes a memory access from an I/O access
/READY	I	Ready – acknowledgement from the addressed memory or I/O that the transfer will be performed

Processor pins

- 80386

Name	Type	Description
INTR	I	Maskable interrupt request – level triggered input sampled during last clock cycle of each instruction, when active the processor acknowledge the interrupt and execute the interrupt service routine
NMI	I	Non-masking interrupt – an edge triggered input which causes an interrupt all the time
/LOCK	O, Z	Bus lock - indicates that other bus masters are not allowed to gain the control of system bus (used for synchronization)
RESET	I	Reset – stops immediately the current execution and restarts the processor
CLK2	I	Double the internal clock

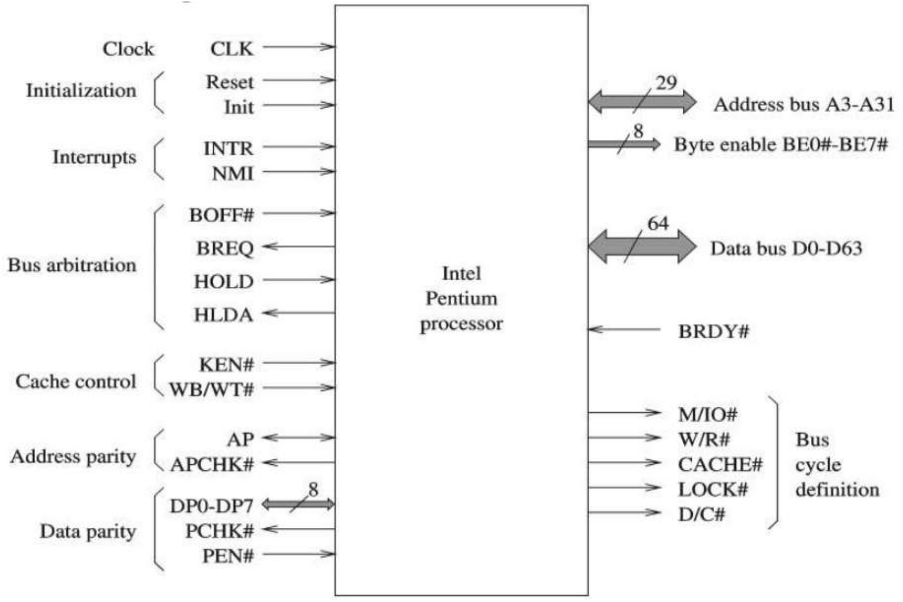
Processor pins

- 80386

Name	Type	Description
PEREQ	I	Coprocessor request (fetch data from coprocessor)
/BUSY	I	Coprocessor busy
/ERROR	I	Coprocessor error
/NA	I	Next address
/BS16	I	Bus size 16 bits
HOLD	I	Bus hold request of another bus master
HLDA	O	Bus hold acknowledgment

Processor pins

- Pentium
  - 32bits processor
  - 64 bits data bus width



Processor pins

- Advanced operations
  - Bus lock (LOCK#)
    - Used in read-modify-write cycle
    - Useful in implementing semaphores
  - Cacheability (CACHE#)
    - Read cycle: indicates internal cacheability
    - Write cycle: burst write-back
  - Cache enable (KEN#)
    - If asserted, the current cycle is transformed into cache line fill
  - WB/ /WT – write-back / write-through

Machine cycles

- Low level external operation of a microprocessor is based on machine cycles
- A machine cycle defines the basic operations that a processor performs to transfer data with external components:
  - memory
  - I/O
- Bus cycles are the external result of executing machine cycles by a microprocessor
- Every interaction between the processor and external components is implemented by a specific machine cycle

Machine cycles

- Every external operation has a dedicated machine cycle, implemented as a bus cycle on microprocessor’s bus:
  - Instruction fetch
  - Memory read
  - Memory write
  - I/O read
  - I/O write
  - Interrupt acknowledge

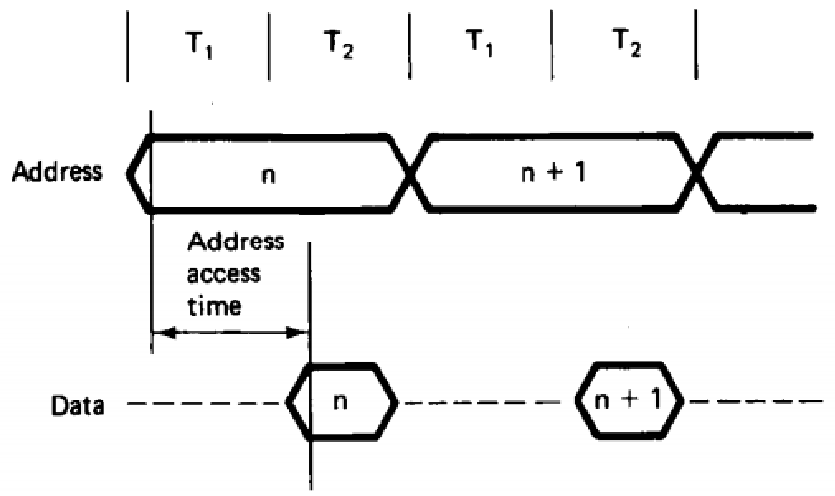
Machine cycles

- Example – 80386 machine cycles encoding

M/ /IO	D/ /C	W/ /R	Bus cycle	Locked
0	0	0	Interrupt acknowledge	Yes
0	0	1	Not possible	-
0	1	0	I/O data read	No
0	1	1	I/O data write	No
1	0	0	Memory code read (Fetch)	No
1	0	1	Processor control - HALT - SHUTDOWN	No
1	1	0	Memory data read	If LOCK
1	1	1	Memory data write	If LOCK

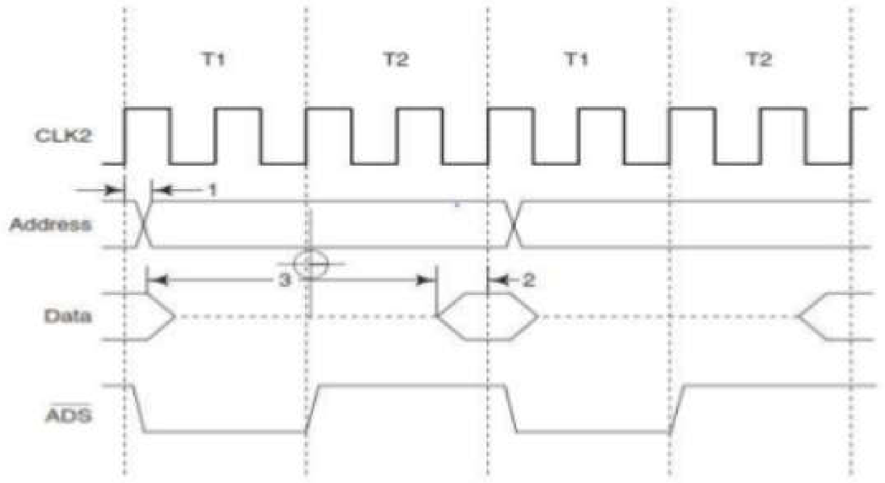
Machine cycles

- Every machine cycle is implemented by a bus cycle on processor external bus
- A bus cycle corresponds to a sequence of events (signal activation and deactivation), called bus protocol
- A bus cycle can be described using time diagrams
  - Typical read/write bus cycles (conceptual)



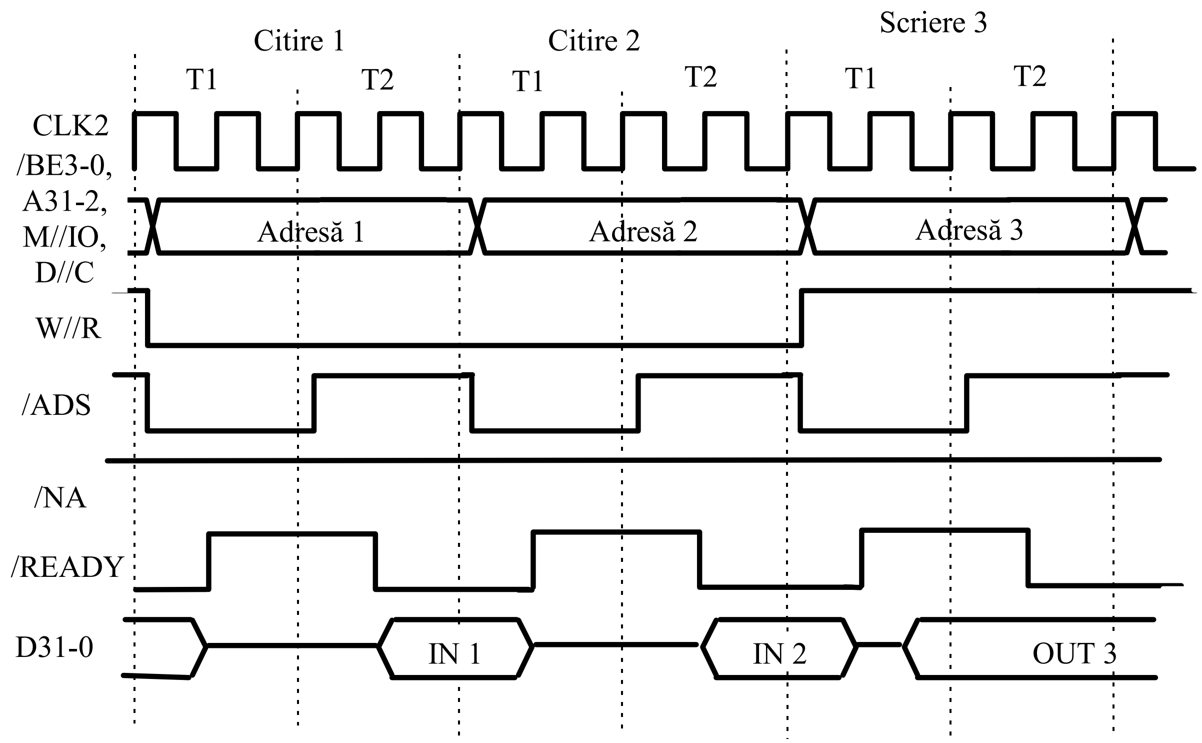
Machine cycles

- A machine cycle is implemented by a specific number of states
  - Called T-States
  - 80386: minimum 2 clock periods, T1-T2, clock period – twice of CLK2
  - Timing diagram for normal cycles



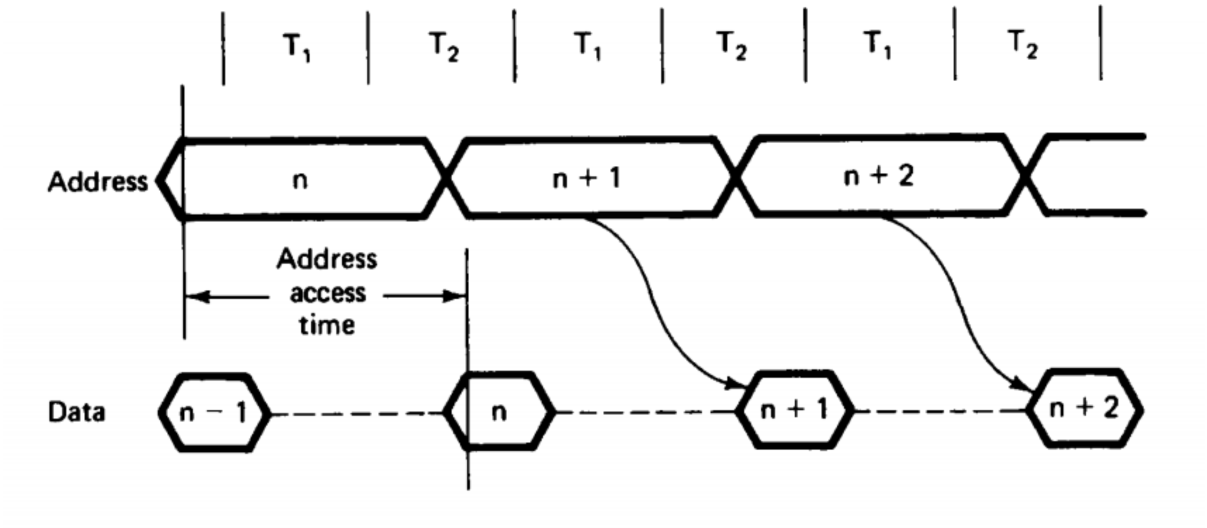
Machine cycles

- Normal cycles (386)



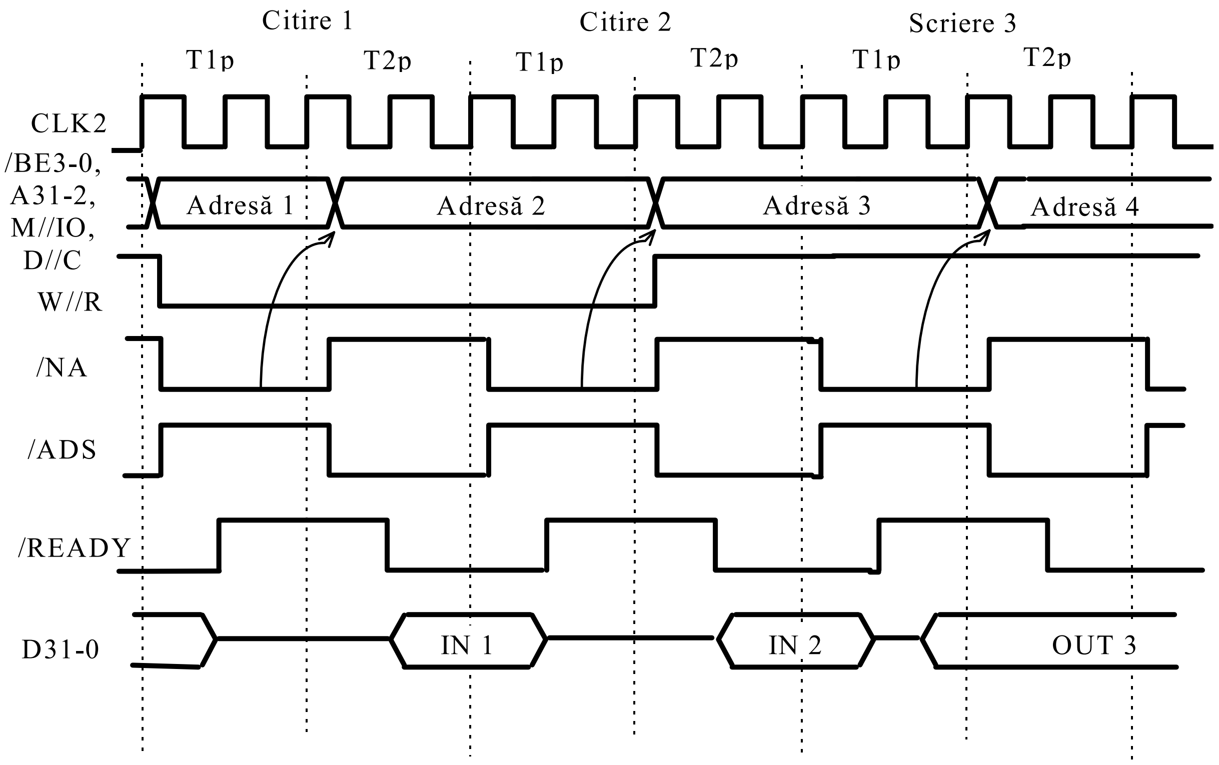
Machine cycles

- Pipelined bus cycles (conceptual)
  - Address for the next cycle is provided in advance, starting in the current cycle

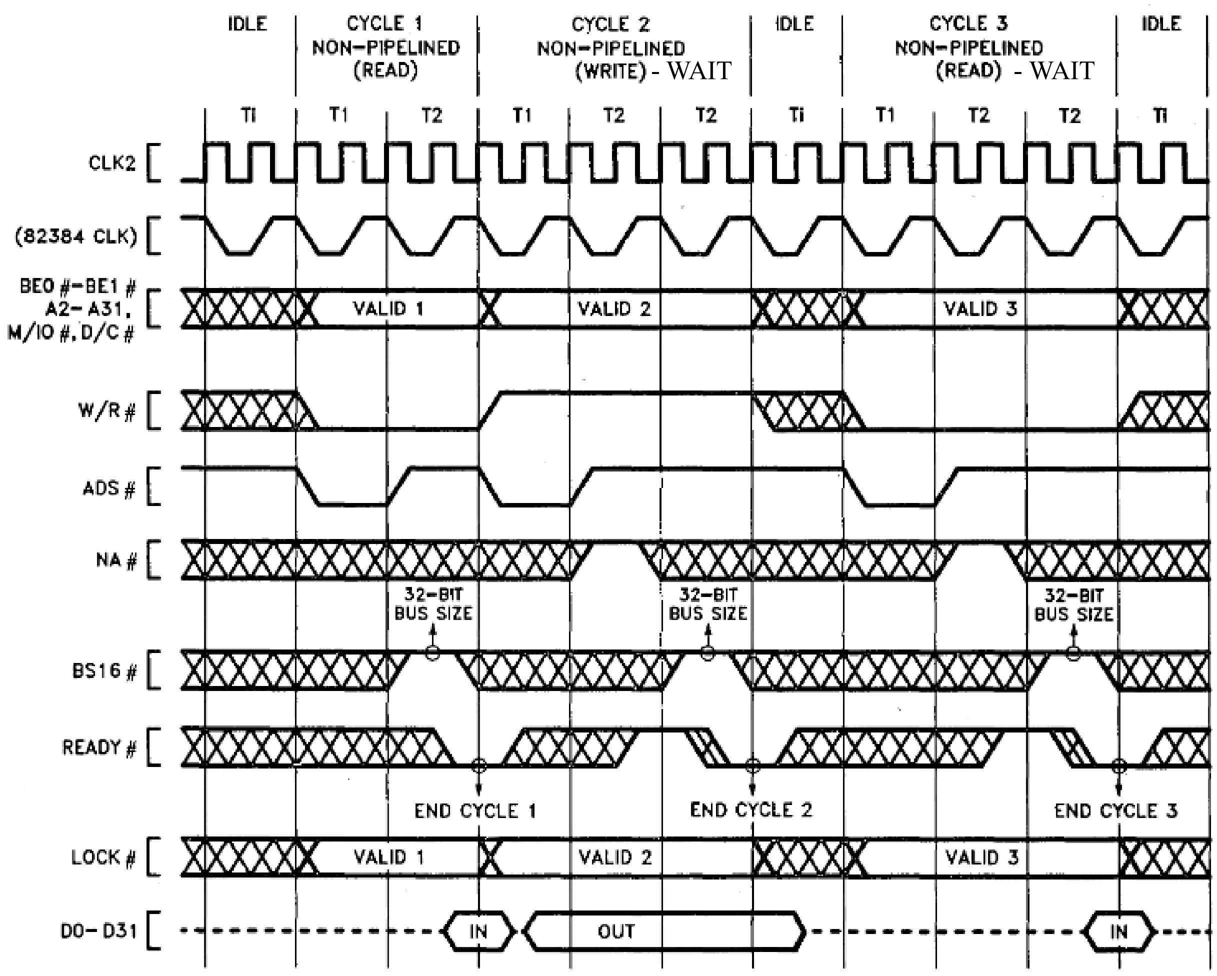


Machine cycles

- Address generated in advance – pipelined cycle (386)



- 386 datasheet
  - Normal cycle
  - Wait states

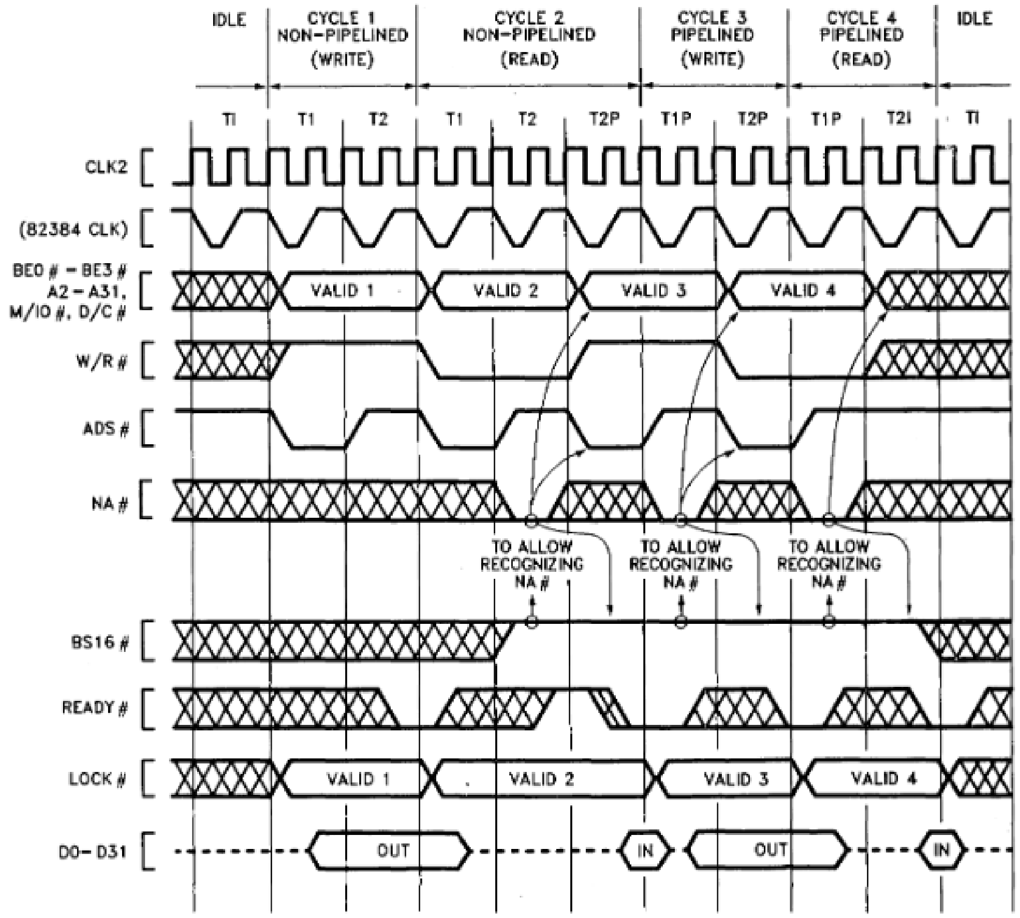

IDLE
CYCLE 1 NON-PIPELINED (READ)
CYCLE 2 NON-PIPELINED (WRITE) - WAIT
IDLE
CYCLE 3 NON-PIPELINED (READ) - WAIT
IDLE
Ti T1 T2 T1 T2 T2 Ti T1 T2 T2 Ti
CLK2
(82384 CLK)
BE0#-BE1# A2-A31, M/IO#, D/C#
VALID 1
VALID 2
VALID 3
W/R#
ADS#
NA#
32-BIT BUS SIZE
BS16#
READY#
END CYCLE 1
END CYCLE 2
END CYCLE 3
LOCK#
D0-D31
IN
OUT

Machine cycles

- Wait states
  - Used by slower components as a mean to ask the processor to wait for the current transaction to be finished
  - Activated according to memory access time

Machine cycles

- 386 datasheet
  - Pipelined cycle



Machine cycles

- How a processor can execute an operation?

```
int a,b;

a = 10;
b = a;
```

Machine cycles

- How many bus/machine cycles a processor will perform to execute one instruction?
  - Every instruction is encoded on several bytes – the machine code of the instruction
  - MOV dest, src

```
MOV AX, 10              ; constant value
; read from a memory location
MOV AX, [BX]            ; [xx] - memory location
; write to a memory location
MOV [BX], AX            ; xx - register content
```

Machine cycles

– MOV dest, src

```
MOV AX, [BX]
```

- Instruction fetch (reads the code of the next instruction to be decoded and executed by the processor)
- Memory read (reads the memory content of the location addressed by BX)

Machine cycles

– MOV dest, src

```
MOV [BX],AX
```

- Instruction fetch (reads the code of the next instruction to be de coded and executed by the processor)
- Memory write (writes the content of the AX register into the memory location addressed by BX)

Machine cycles

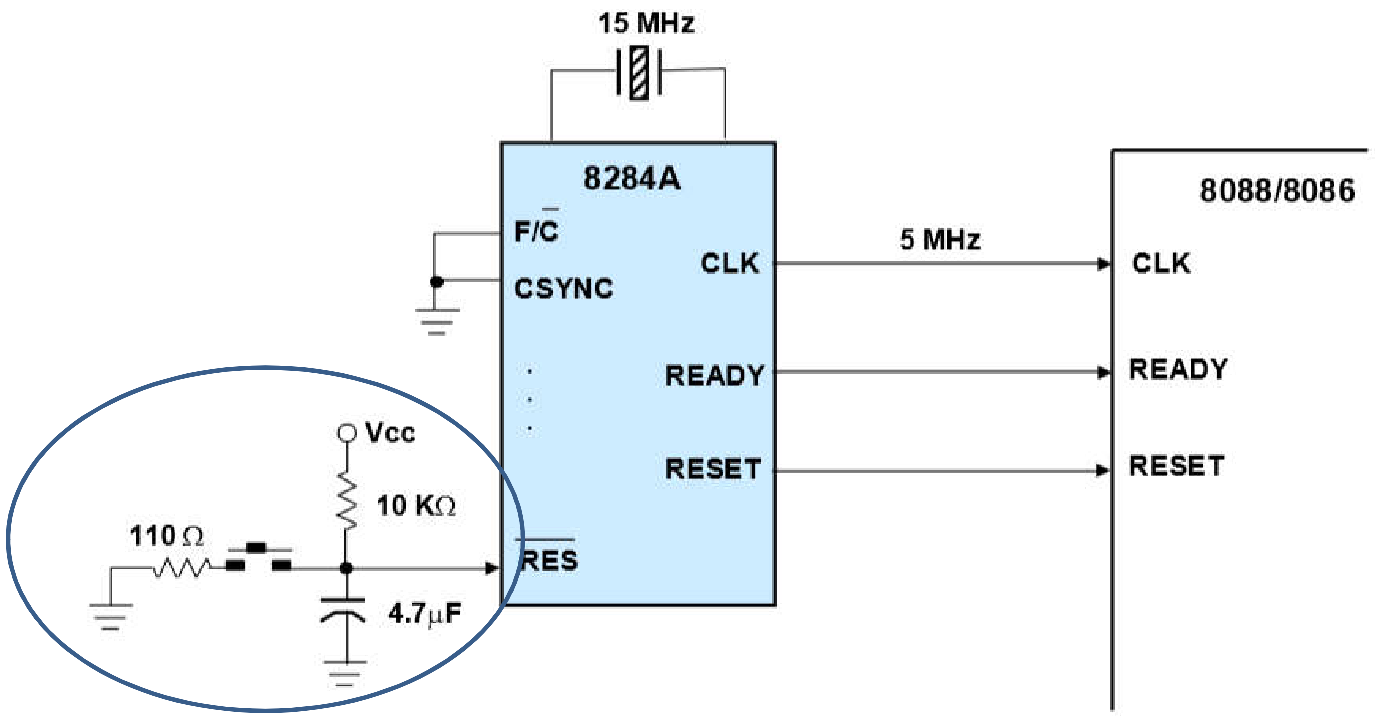
- The execution of one instruction is achieved by a sequence of several machine cycles:
  - Instruction fetch,
  - Read operands,
  - Internal execution
  - Write results
- A machine cycle can be extended by inserting wait states, a state is a clock cycle or clock period.
  - When the memory is slow

Machine cycles

- Clock and reset generator
  - Generates clock signals
    - Processor
    - Specialized circuits
  - Generates READY signal
    - Synchronized with clock signal
    - When inactive the current machine cycle will wait for a slower component involved in current transaction
    - Will add wait states to current machine cycle
  - Generated RESET signal
    - Synchronized with the clock

Machine cycles

- Clock and reset generator



Processor startup

- Reset
  - Input pin – hardware reset
  - When active the processor stops immediately its execution
  - Reset signal is synchronized with clock signal
    - Reset generator module
  - It should be kept active for minimum n clock cycles (e.g. n = 4 - 8086)

Processor startup

- Reset
  - After reset internal registers are initialized with predefined values:

Register	Content
FLAGS	0000H
IP	0000H
CS	FFFFH
DS	0000H
SS	0000H
ES	0000H
Instruction buffer	Empty

Processor startup

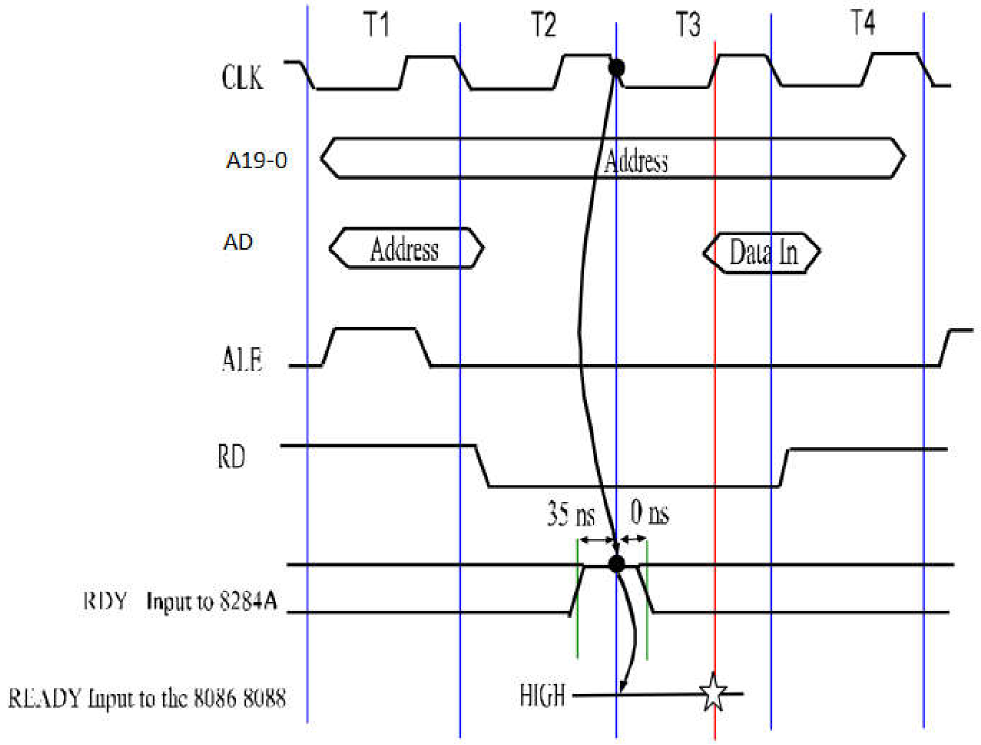
- Reset
  - After reset the processor will start executing code from logical address FFFFH:0000H
    - Real mode
  - ROM should be mapped at this address
    - Physical address FFFF0H
  - Initialization and boot loader code is executed from ROM

Machine cycles

- What happens whether the memory is not fast enough to provide or store data to/from a processor?

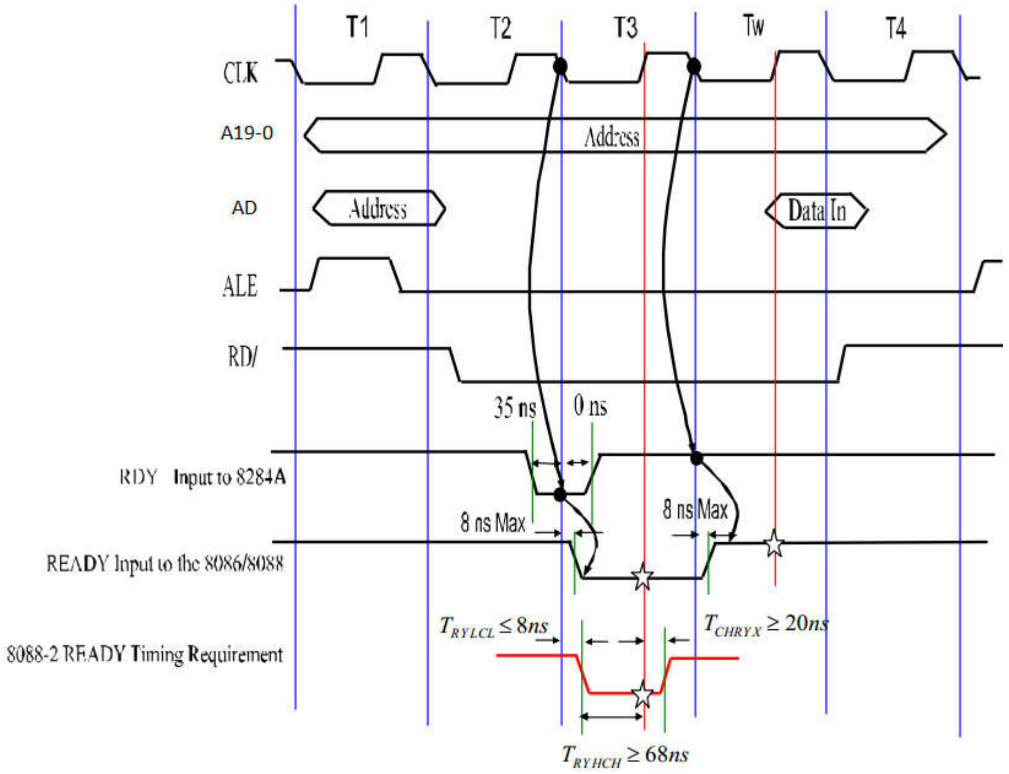
Machine cycles

- Zero wait states



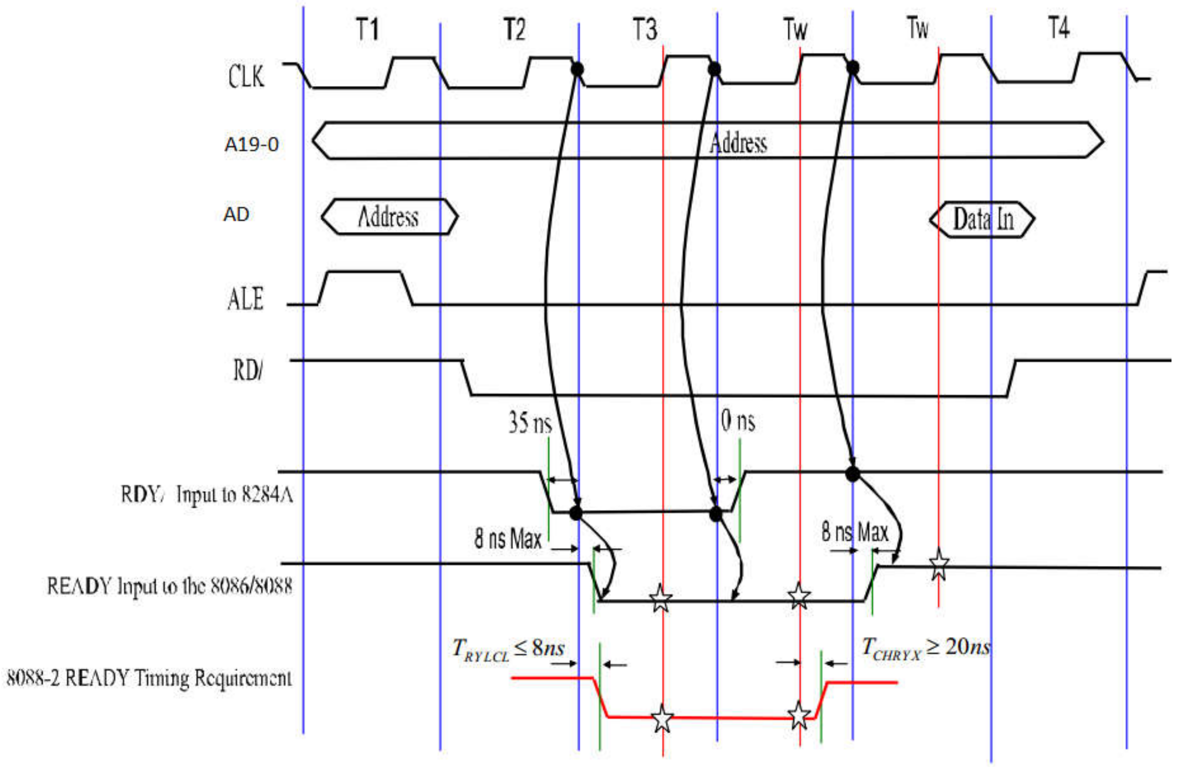
Machine cycles

- One wait state



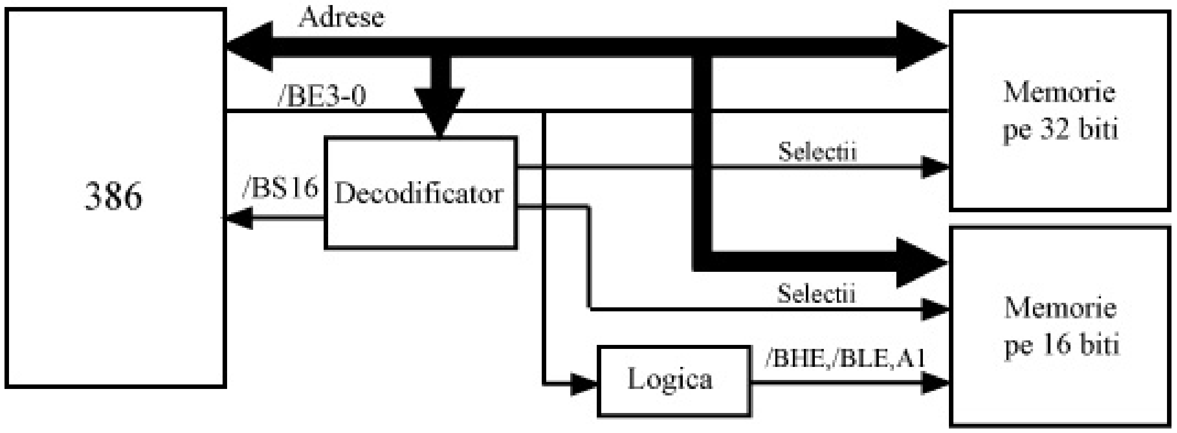
Machine cycles

- Two wait states



Data transfer

- 80386
  - Transfers on 32 bits and 16 bits

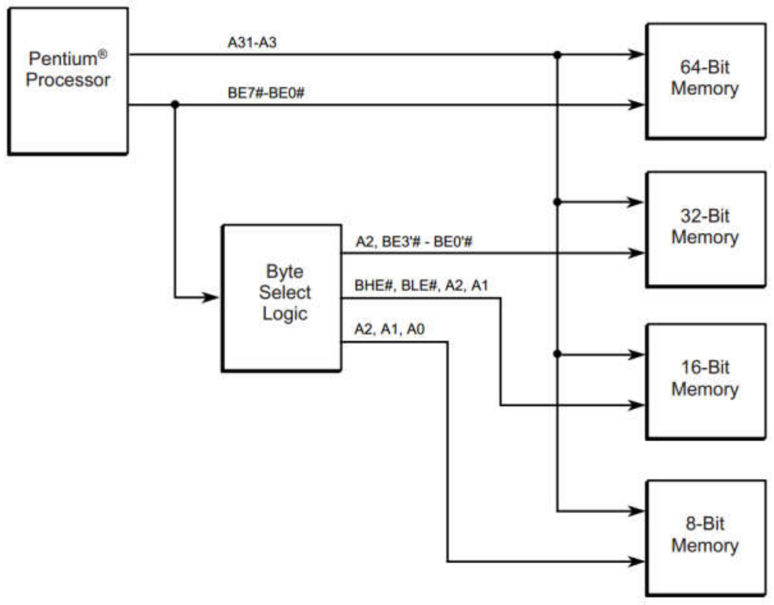


Data transfer

- Exercise
  - Design the memory map and memory decoder for a 32 bits microprocessor system with 32 address lines using the following memory requirements:
    - 16MB EEPROM, using 4M x 16 bits memories
    - 1MB SRAM, using 256K x 32 bits circuits
    - 1GB DRAM, using 128M x 8 bits memories

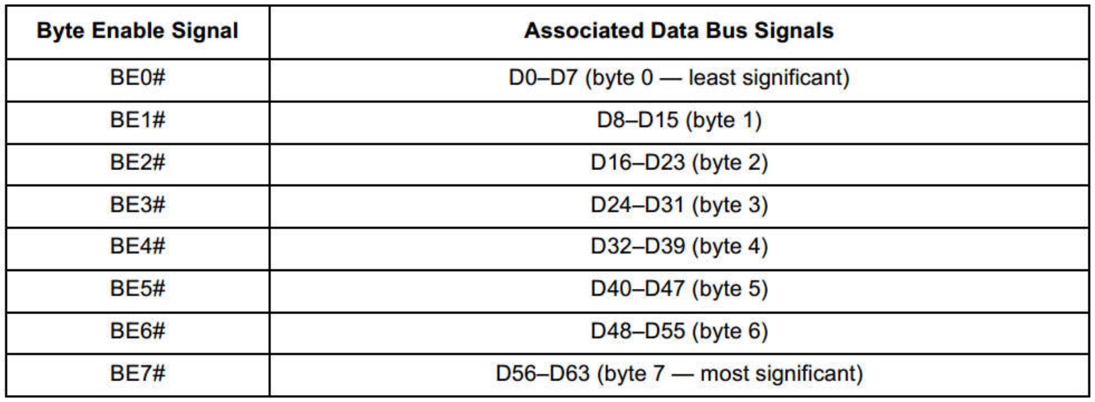
Data transfer

- Bytes enable
  - 32 bits uP (e.g. 80386) does not contain address connections A1 and A0 because these have been encoded as the byte enable signals (BE3-0).
  - 64 bits uP does not contain address connections A2-A0 because these have been encoded as the byte enable signals (BE7-0).



Data transfer

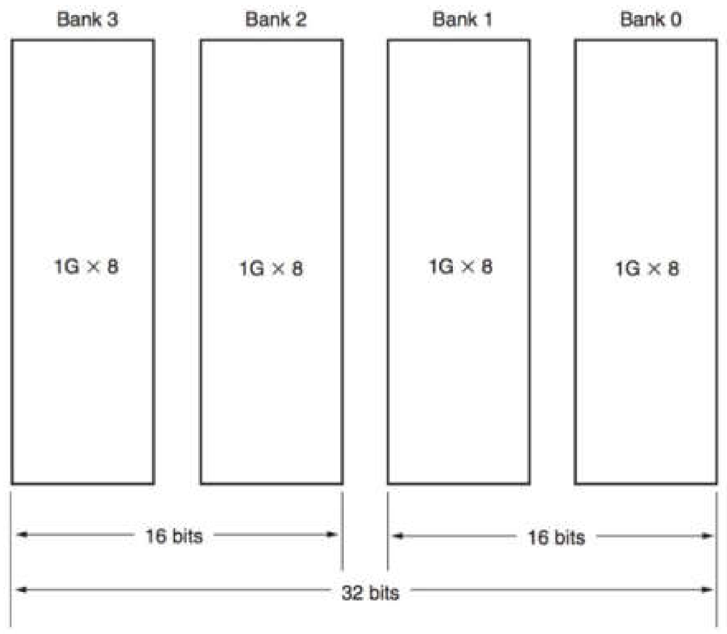
- 32 bits
  - Data Bus: D_{31} - D_0
    - Bi-directional Bus
    - Dynamic Bus size : BS16 input
    - BE_0 : D_0 - D_7
      BE_1 : D_8 - D_{15}
      BE_2 : D_{16} - D_{23}
      BE_3 : D_{24} - D_{31}
- 64 bits – Data Bus D63-D0



Byte Enable Signal	Associated Data Bus Signals
BE0#	D0–D7 (byte 0 — least significant)
BE1#	D8–D15 (byte 1)
BE2#	D16–D23 (byte 2)
BE3#	D24–D31 (byte 3)
BE4#	D32–D39 (byte 4)
BE5#	D40–D47 (byte 5)
BE6#	D48–D55 (byte 6)
BE7#	D56–D63 (byte 7 — most significant)

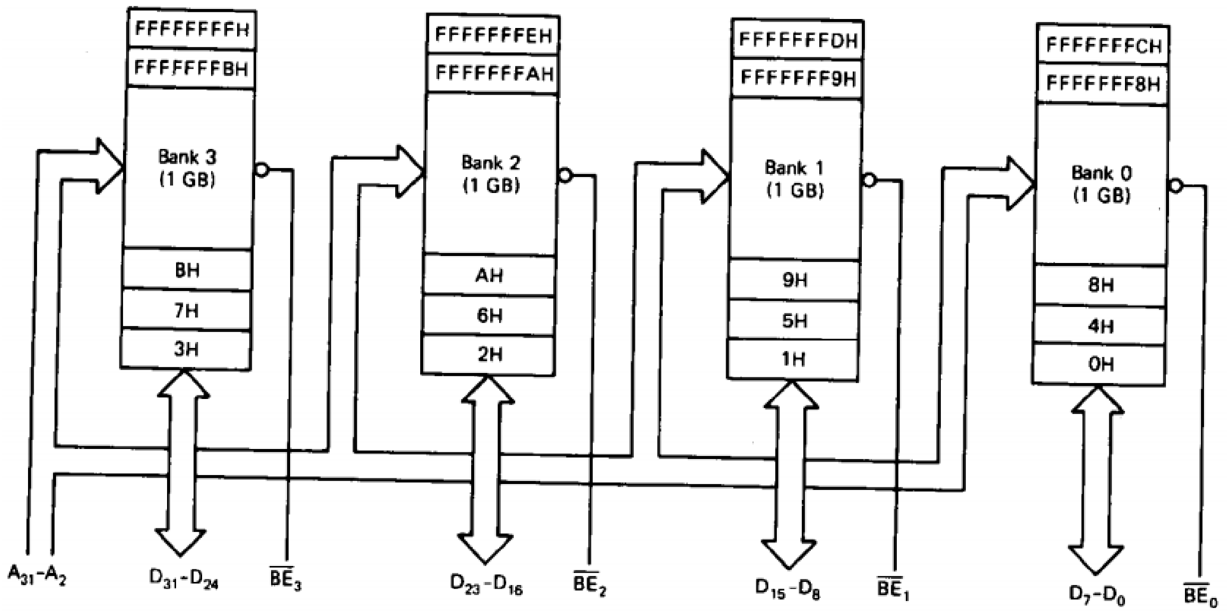
Data transfer

- 32 bits data bus



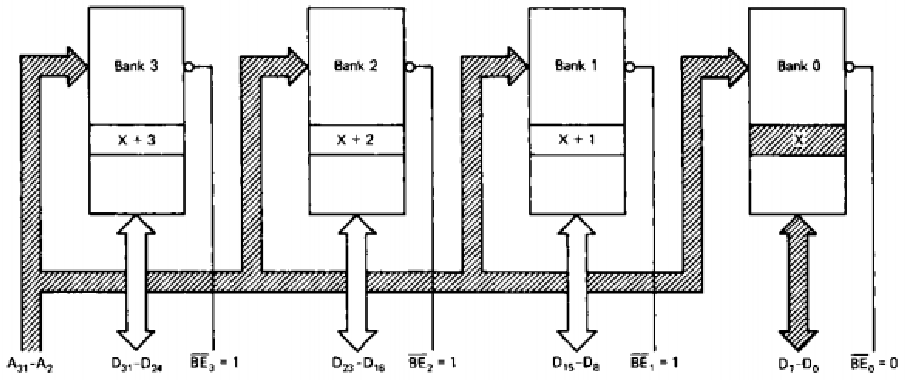
Data transfer

- Hardware organization of physical address space (32 bits)
  - 32 bits memory block



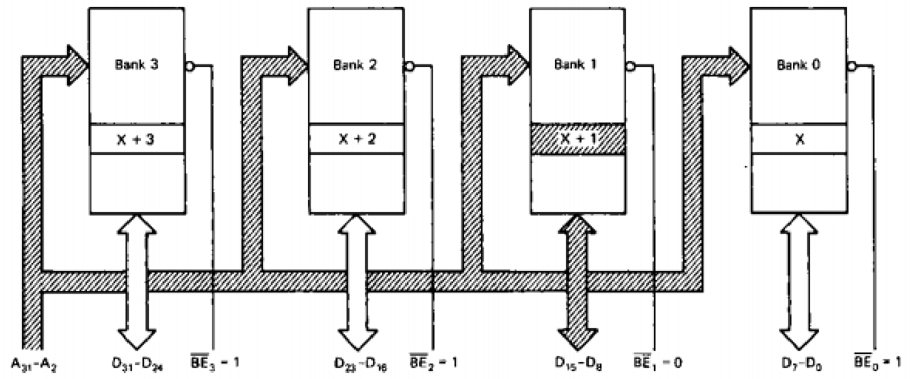
Data transfer

- 1 byte
  - Address 4xk



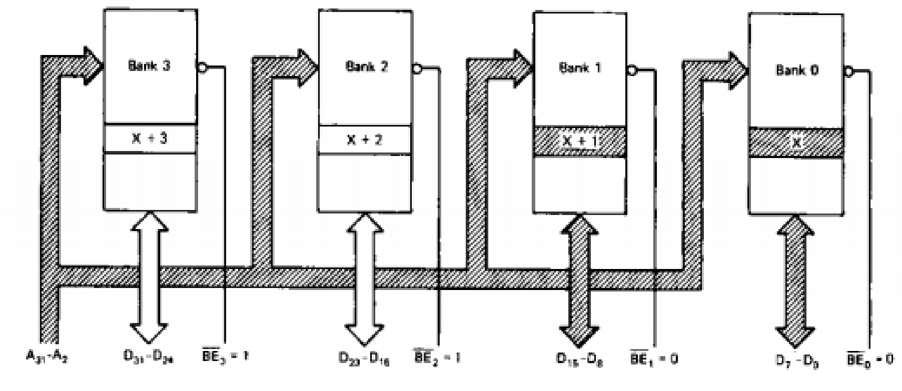
(a)

  - Address 4xk+1



Data transfer

- 1 word
  - – Address 4xk

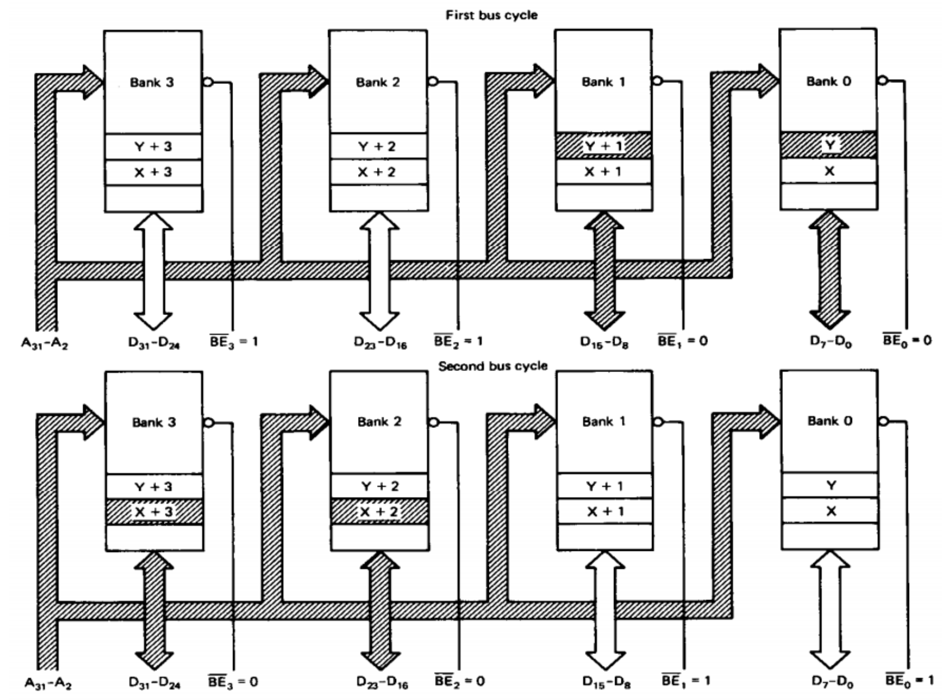


(c)

- 1 double word
  - – Address 4xk

Data transfer

- Misaligned double-word data transfer
  - Address 4xk+2 – the processor will perform 2 bus cycles



Pipeline

- Most instructions have common operations
  - Fetch instruction code from memory
  - Decode instruction code
  - Select operand registers
  - Use ALU for arithmetic and logic operations

Pipeline

- For each instructions the processor performs several tasks:
  - Instruction fetch (IF)
  - Instruction decode (ID)
  - Execution (EX)
  - Memory access (MEM)
  - Write-back (WB)
- However, each instruction uses different subsets of existing functional units
- Hence the instructions need various times to complete
  - Some instructions are faster than others

Pipeline

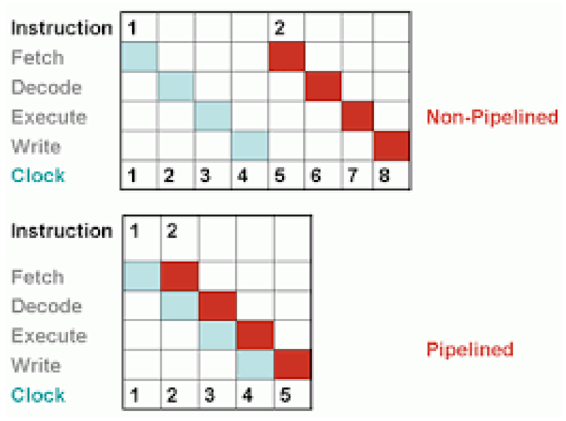
- If the processor executes instructions internally in one clock cycle, the operating frequency is established based on the slowest instruction
- Sequential execution of all tasks required by each instruction is called single-cycle execution

Pipeline

- Single cycle data path is not efficient in time
  - Clock cycle time is determined by the instruction taking the longest time
  - Variable clock cycle time is too complicated
- Alternative design approaches
  - Multiple clock cycles per instruction

Pipeline

- Pipelining is a microprocessor implementation technique in which multiple instructions are overlapped in execution
- One instruction requires several clock cycles to execute



Pipeline

- The pipeline is divided in stages
- Each stage completes a part of an instruction in parallel with the rest of the stages
- The stages are connected one to the next to form a pipe
  - Outputs of one stage are inputs for the next stage
  - The instructions enter at one end, progress through the stages, and exit at the other end

Pipeline

- Pipelining
  - Multiple tasks operating simultaneously using different resources
  - It does not reduce latency of single instruction execution, it increase the throughput of entire workload (several instruction which feed the pipeline)
  - Pipeline rate is limited by slowest pipeline stage
    - Unbalanced lengths of pipeline stages reduces speedup

Pipeline

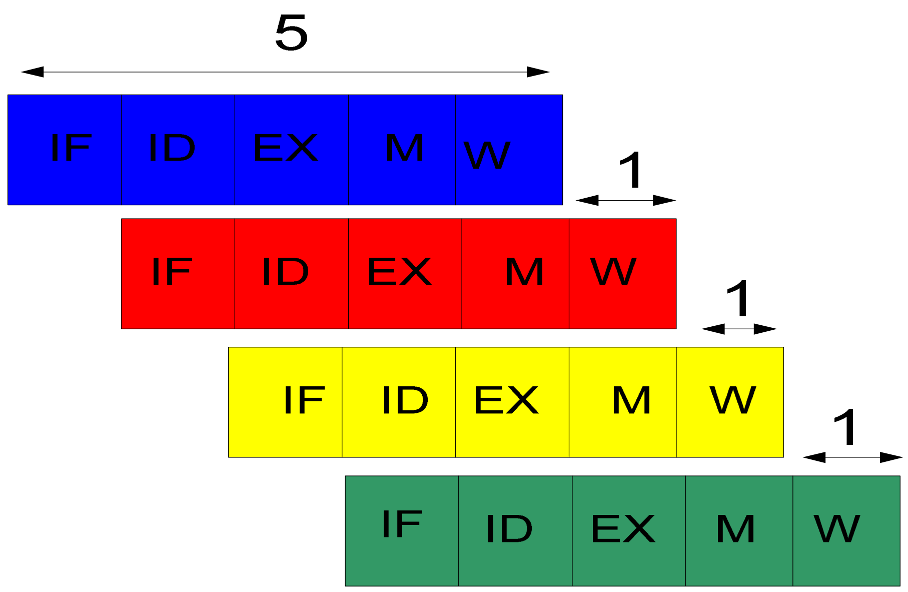
- Pipelining speedup
  - The potential speedup is determined by the number of the stages
  - The effective speedup is limited by
    - Time needed to fill the pipeline with instructions
    - Time needed to finish last instructions
    - Stalls for dependencies
    - Cleaning for branches

Pipeline

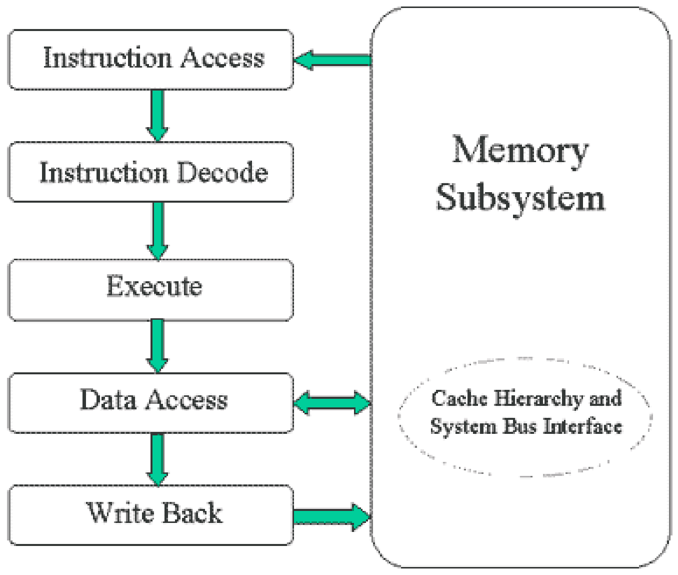
- Pipelining implementation
  - Instruction execution tasks becomes pipeline stages:
    - Instruction fetch (IF)
    - Instruction decode (ID)
    - Execution (EX)
    - Memory access (MEM)
    - Write-back (WB)
  - Each stage performs its operation in one clock cycle
  - The clock speed is determined by the slowest stage

Pipeline

- Five stage pipelining – concept



Four Pipelined Instructions



Pipeline

- Instruction Fetch (IF) stage
  - Is responsible for reading the requested instruction from memory.
  - The instruction and the program counter (which is incremented to the next instruction) are stored in the IF/ID pipeline register as temporary storage so that may be used in the next stage at the start of the next clock cycle.
- Instruction Decode (ID) stage
  - Is responsible for decoding the instruction and sending out the various control lines to the other parts of the processor.
  - The instruction is sent to the control unit where it is decoded and the registers are fetched from the register file.

Pipeline

- Execution (EX) stage
  - Is where any calculations are performed.
  - The main component in this stage is the ALU.
  - The ALU is made up of arithmetic, logic capabilities.
- Memory and IO (MEM) stage
  - Is responsible for storing and loading values to and from memory.
  - It also responsible for input or output from the processor.
  - If the current instruction is not of Memory or IO type than the result from the ALU is passed through to the write back stage

Pipeline

- Write-Back (WB) stage
  - Is responsible for writing the result of a calculation, memory access or input into the register file.

Pipeline

- Advantages:
  - More efficient use of processor's resources
  - Decrease execution time of blocks of instructions
  - Support for ILP

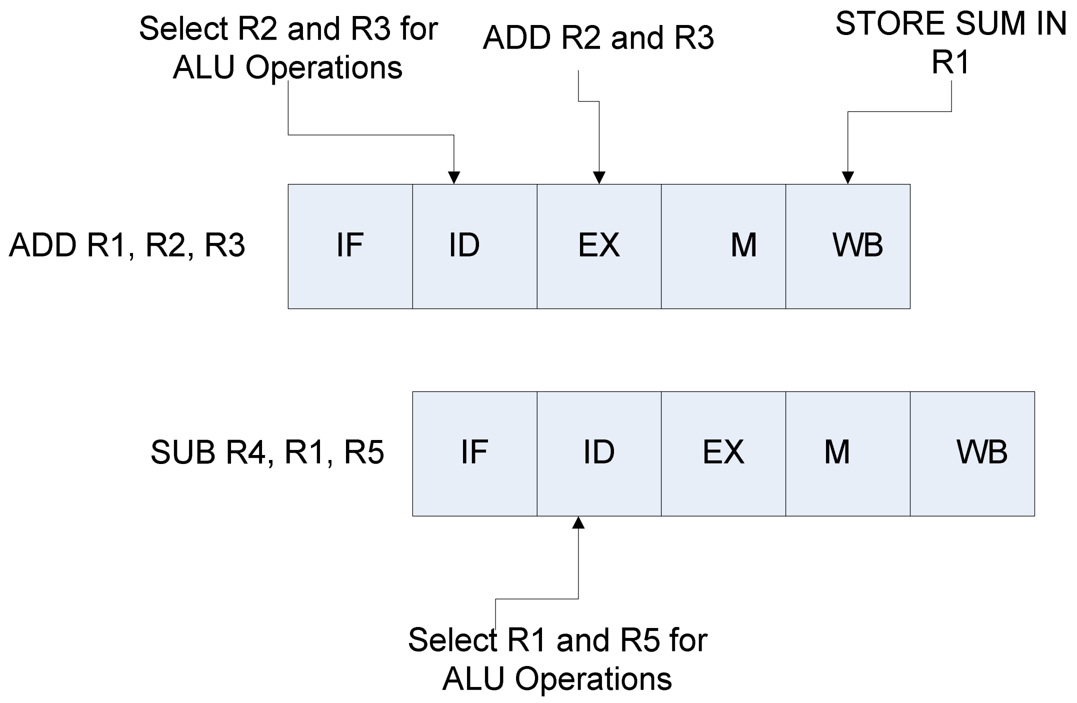
- Disadvantages:
  - Pipelining involves adding hardware to the chip
  - Inability to continuously run the pipeline at full speed because of pipeline hazards which disrupt the smooth execution of the pipeline.
  - Increase execution time of one instruction

Pipeline

- Situations that prevent starting the next instruction in the next cycle are called hazards
  - Structural Hazards – two instructions need to access the same resource
  - Data Hazards – an instruction uses the result of the previous instruction
  - Control Hazards – the location (address) of an instruction depends on previous instruction

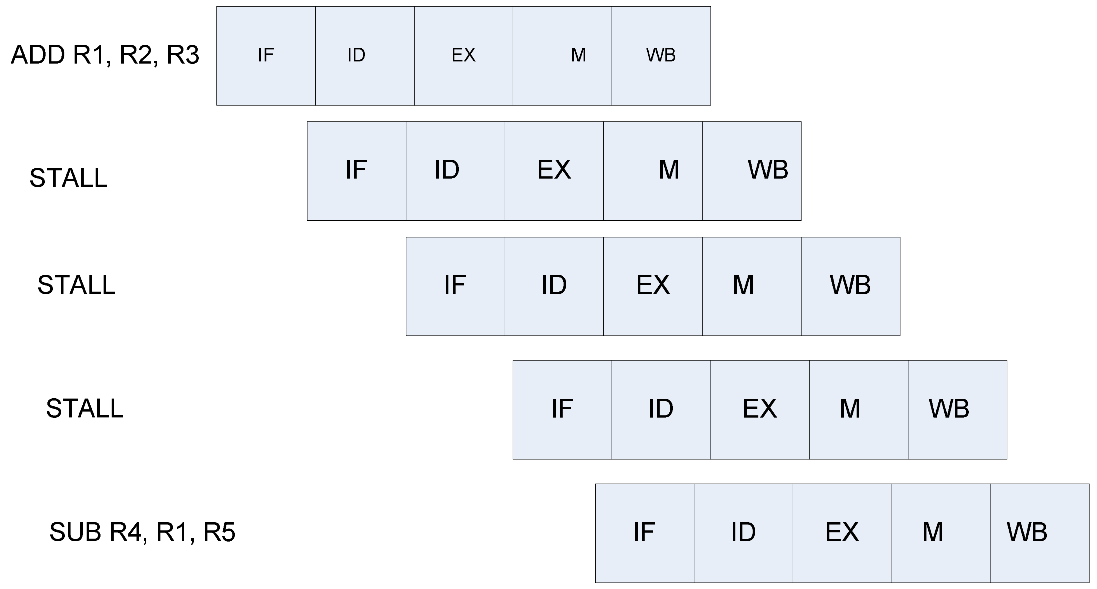
Pipeline

- Data hazards



Pipeline

- Data hazards
  - Stalling - involves halting the flow of instructions until the required result is ready to be used. However stalling wastes processor time by doing nothing while waiting for the result

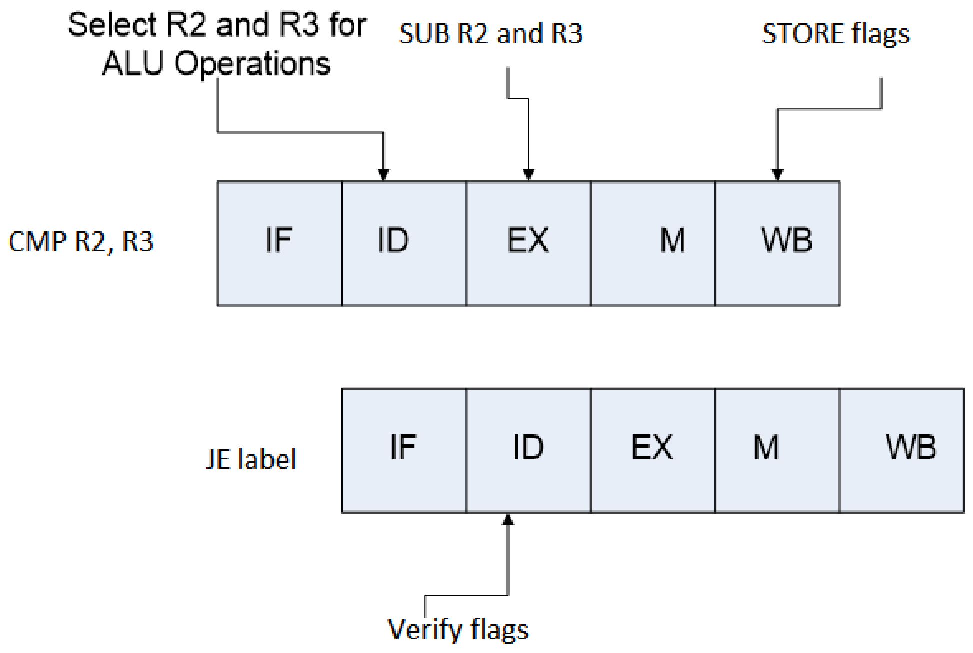


Pipeline

- Control hazards
  - Branch determines flow of control
    - Fetching next instruction depends on branch outcome
    - Pipeline can’t always fetch correct instruction
      - Still working on ID stage of branch

Pipeline

- Control hazards



Pipeline

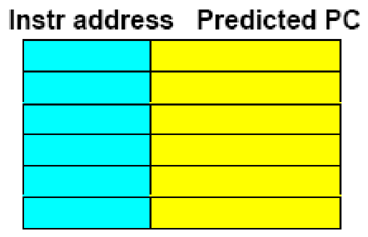
- Control hazards solutions
  - Stalls
  - Branch prediction
    - Longer pipelines cannot determine branch outcome early
      - Stall penalty becomes unacceptable
    - Predict: guess one direction then back up if wrong
    - Predict outcome of branch
      - Only stall if prediction is wrong
      - Empty the pipeline
    - Avoid branches
      - Conditional instructions (e.g. CMOV)
      - Predicative instructions

Pipeline

- Static branch prediction
  - Based on typical branch behavior
  - Example: loop and if-statement branches
    - Predict backward branches taken
    - Predict forward branches not taken
- Dynamic branch prediction
  - Hardware measures actual branch behavior
    - record recent history of each branch
  - Assume future behavior will continue the trend
    - When wrong, stall while re-fetching, and update history

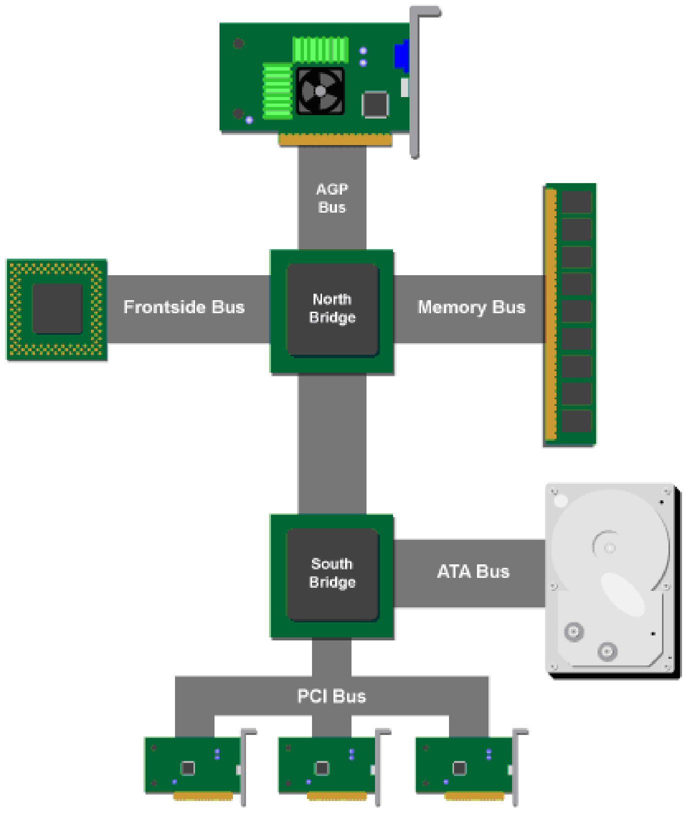
Pipeline

- Branch Target Buffer – BTB
  - Cache memory which stores target addresses of branch instructions
  - Predicted addresses are decided at code execution
  - Additional flags encoding branch address probability



System architecture

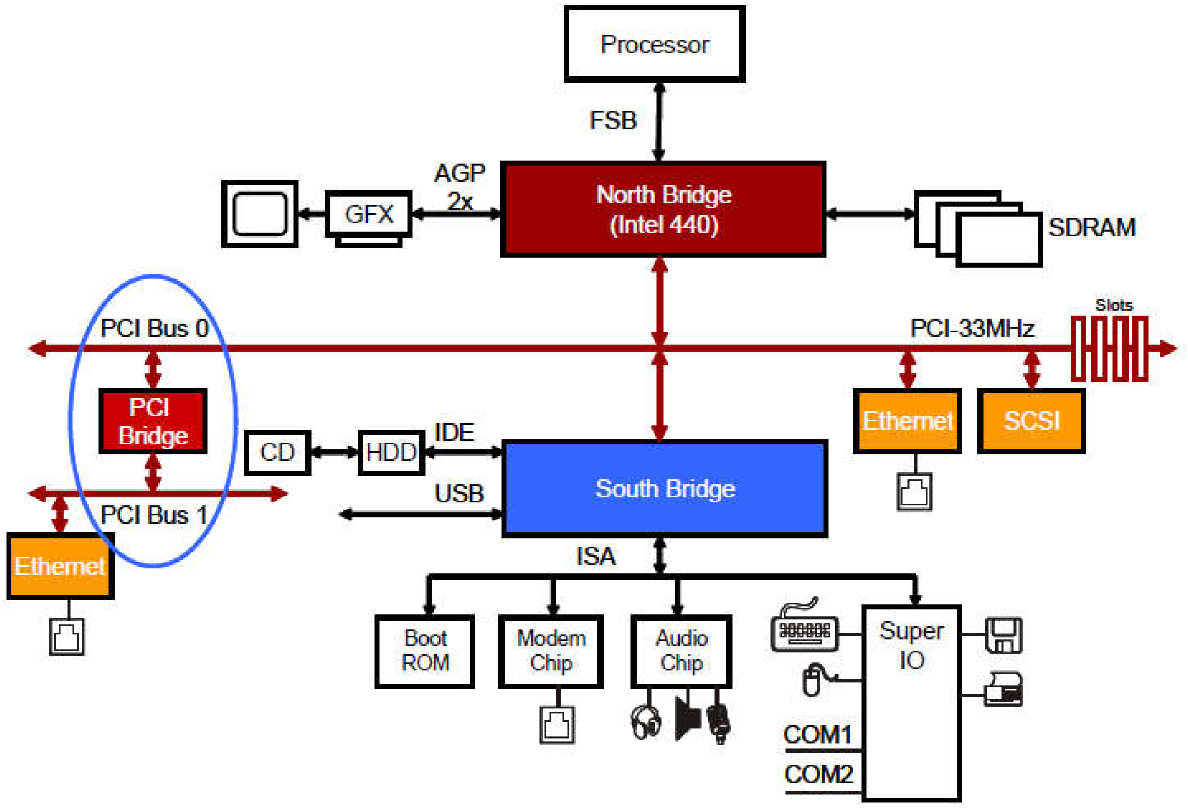
- FSB
- Memory bus
- Chipset
- I/O bus



System architecture

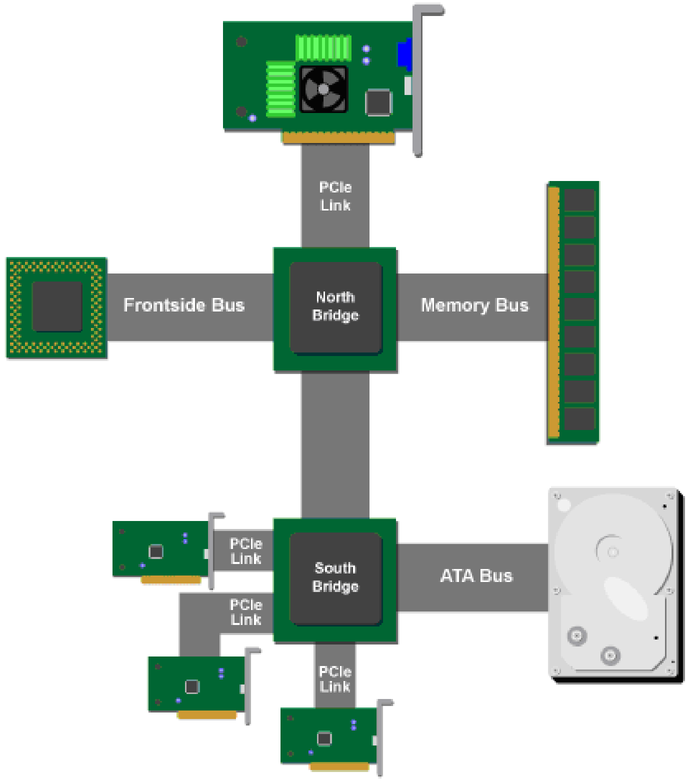
- Chipsets
  - Designed for use with a specific family of processors
  - Northbridge: connects the CPU to high-speed components like RAM, video card
  - Southbridge: connects to slower peripheral devices like PCI slots, USB, IDE, ...

System architecture

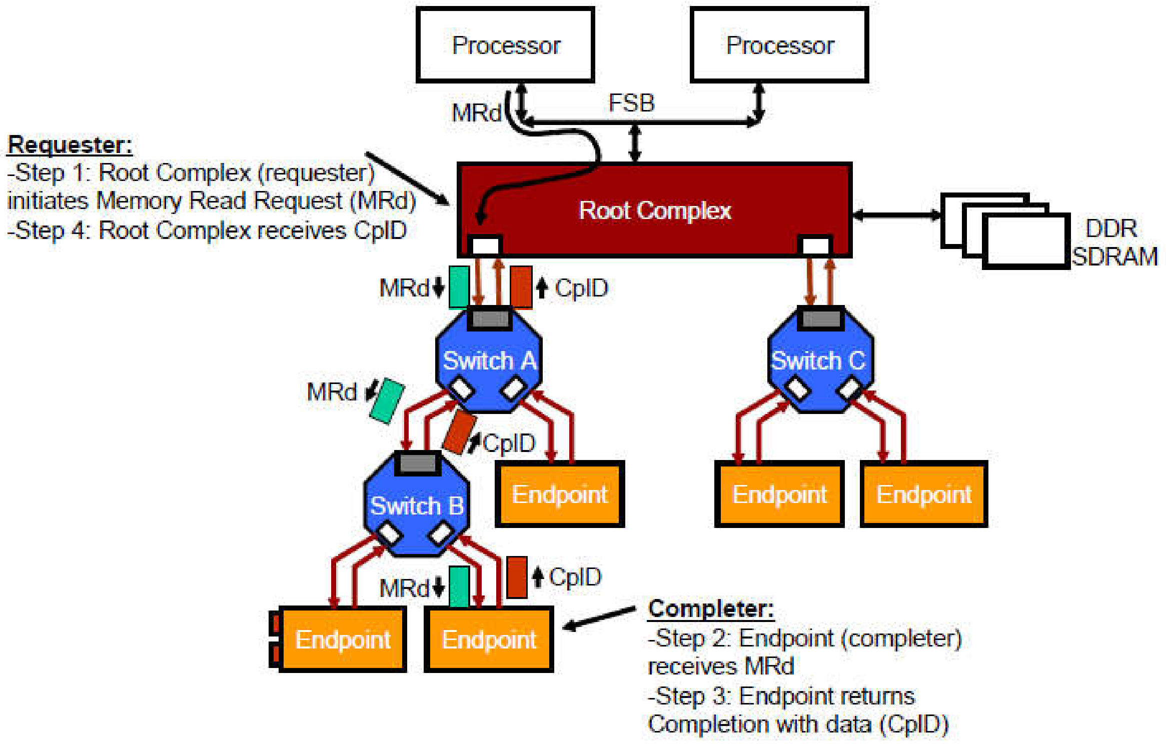


System architecture

- PCI express

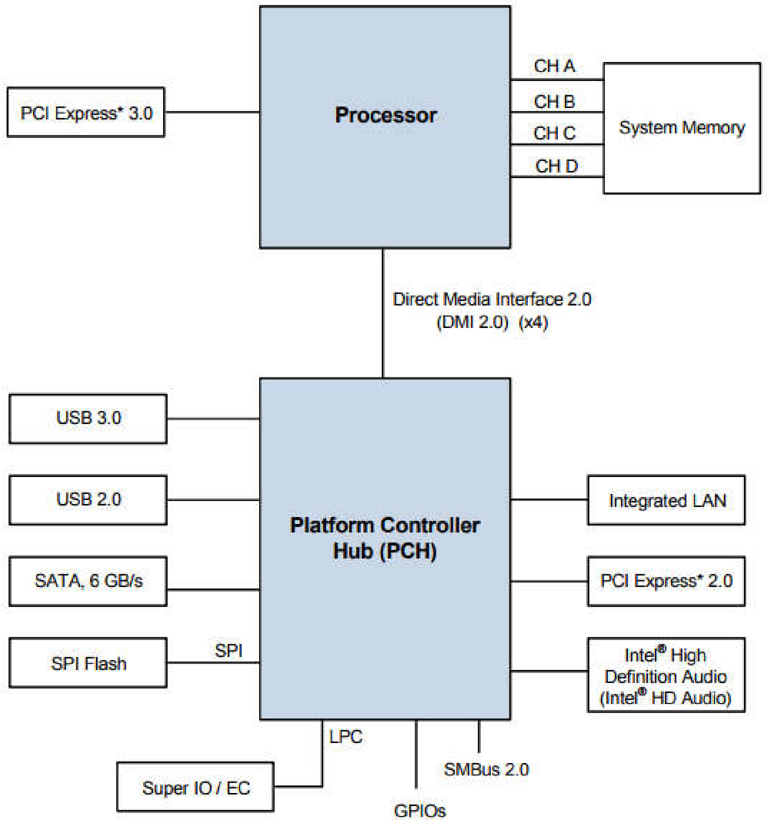


System architecture


Processor
Processor
MRd
FSB
Requester:
-Step 1: Root Complex (requester) initiates Memory Read Request (MRd)
-Step 4: Root Complex receives CplD
Root Complex
DDR SDRAM
MRd
CplD
Switch A
Switch C
MRd
CplD
Switch B
Endpoint
Endpoint
Endpoint
MRd
CplD
Endpoint
Endpoint
Completer:
-Step 2: Endpoint (completer) receives MRd
-Step 3: Endpoint returns Completion with data (CplD)

System architecture

- iCore7



Interconnect

- FSB

Figure 3. Shared Front-side Bus, up until 2004

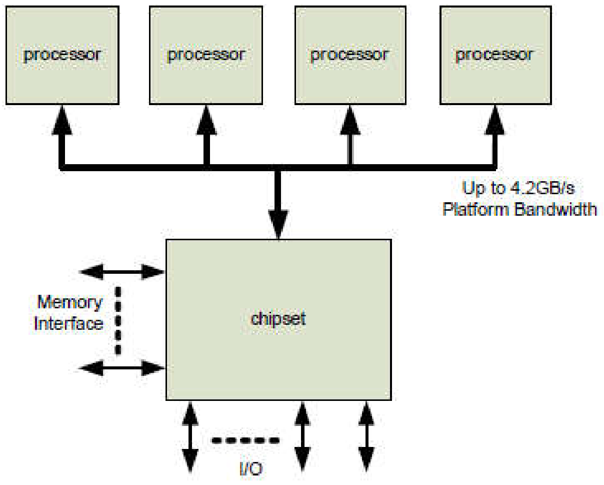
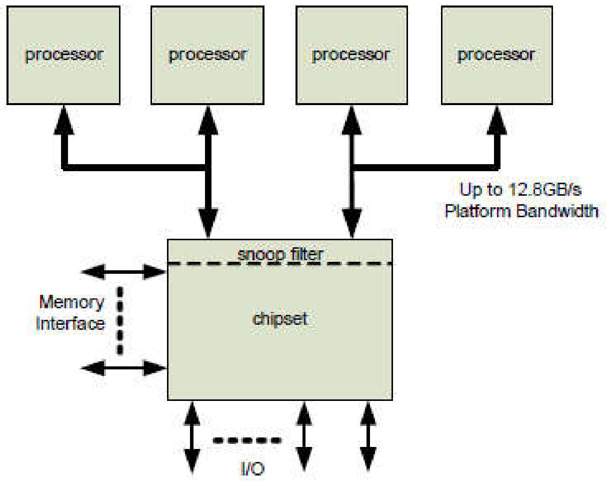


Figure 4. Dual Independent Buses, circa 2005



Interconnect

- DHSI, QPI
  - Point-to-point

Figure 5. Dedicated High-speed Interconnects, 2007

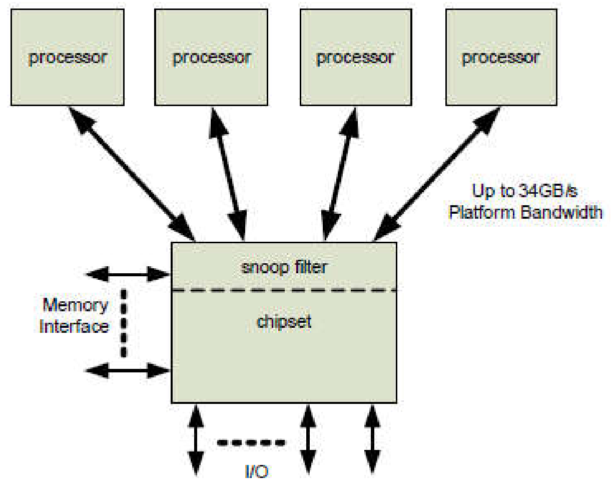
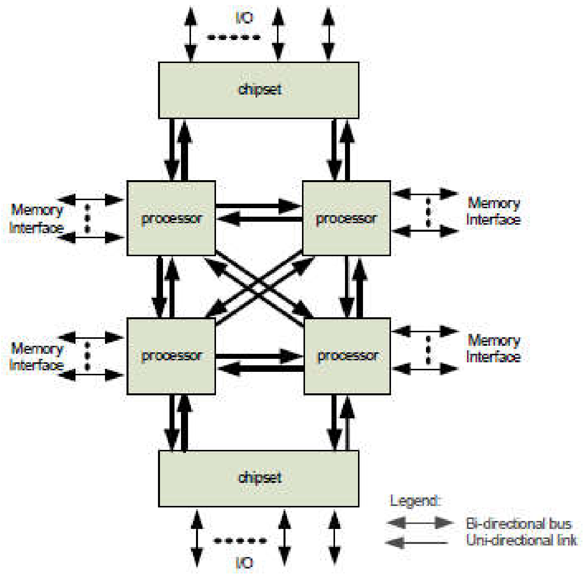


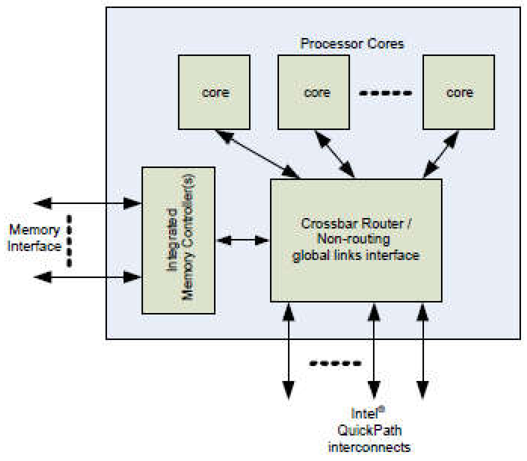
Figure 6. Intel® QuickPath Interconnect



Interconnect

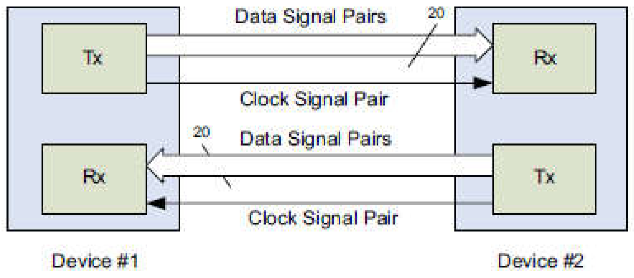
- QPI
  - High-speed, packetized, point-to-point interconnect
  - Distributed shared memory architecture (NUMA)

Figure 7. Block Diagram of Processor with Intel® QuickPath Interconnects



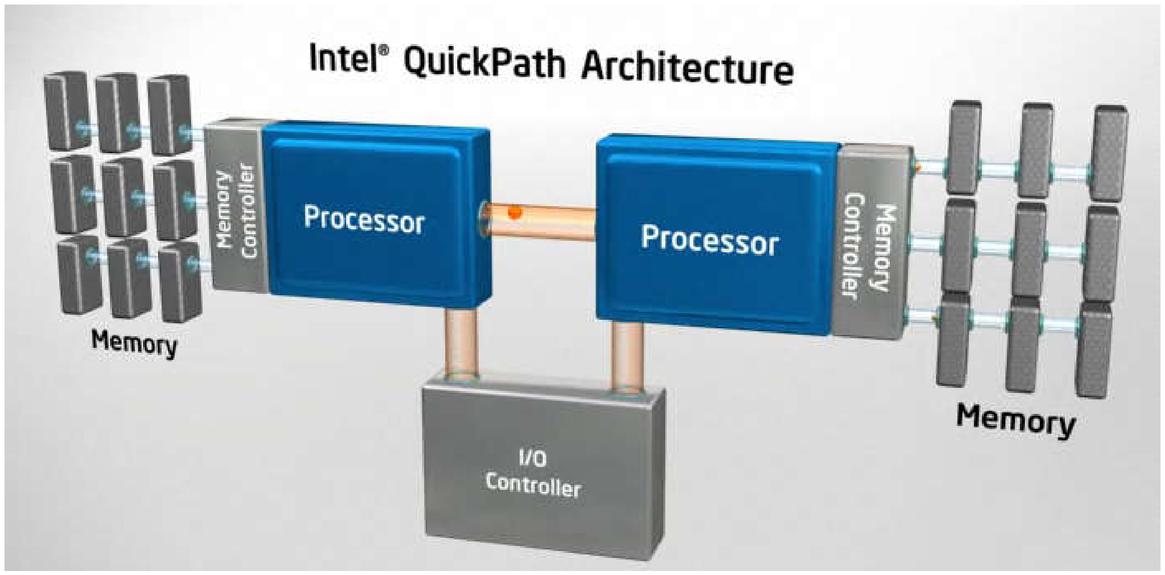
Processor Cores
core
core
core
Memory Interface
Integrated Memory Controller(s)
Crossbar Router / Non-routing global links interface
Intel® QuickPath interconnects

Figure 9. Physical Layer Diagram



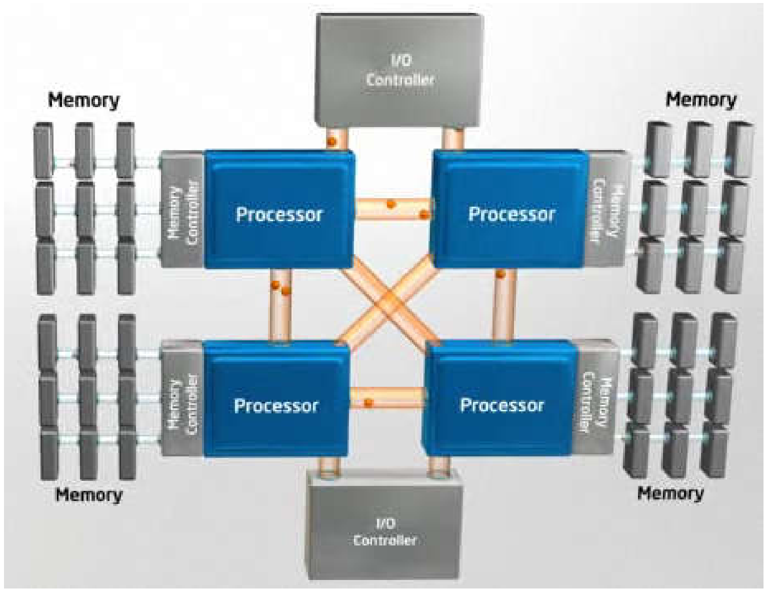
Interconnect

- Quick Path Interconnect
  - iCore
    - NUMA



Interconnect

- Quick Path Interconnect
  - FSB - ~13GBps
  - QPI - ~26GBps

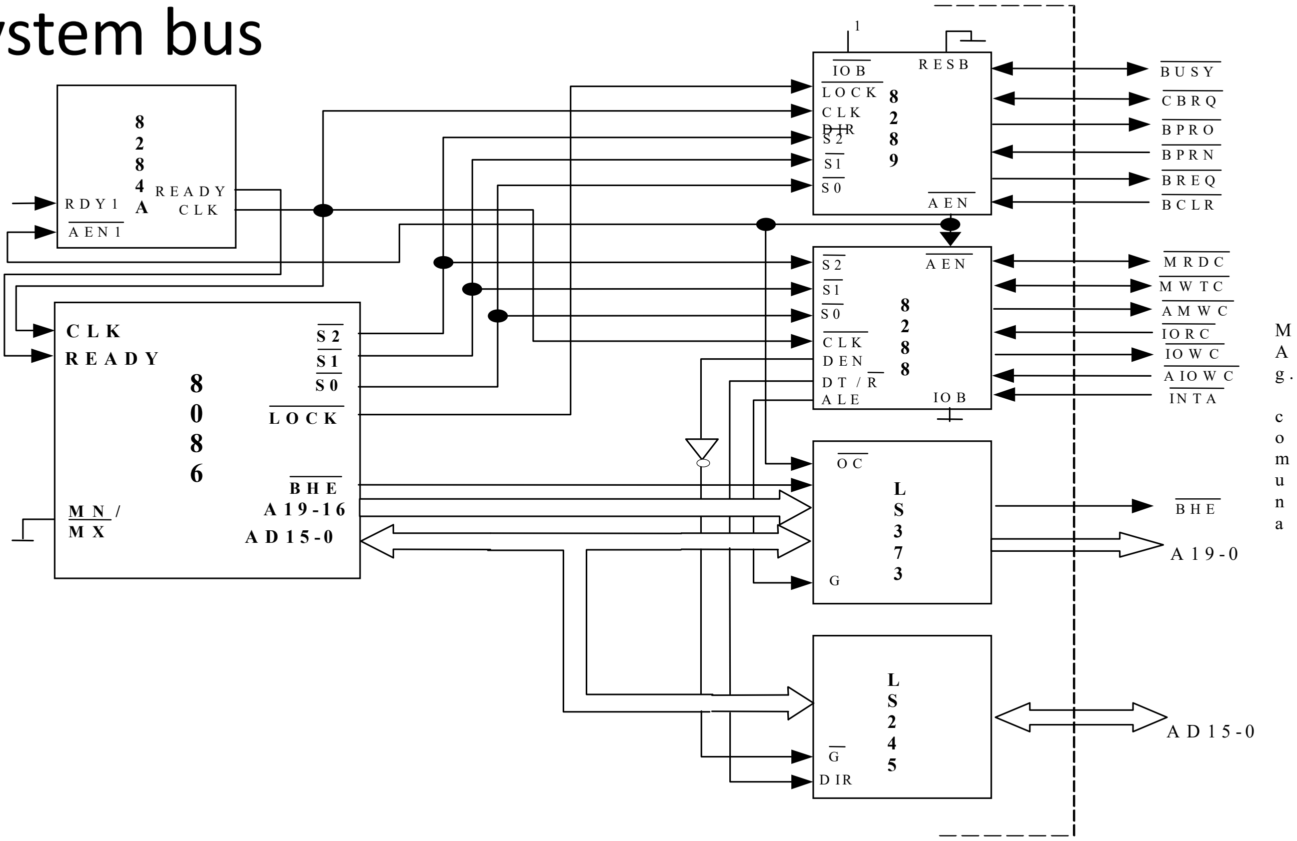


Summary

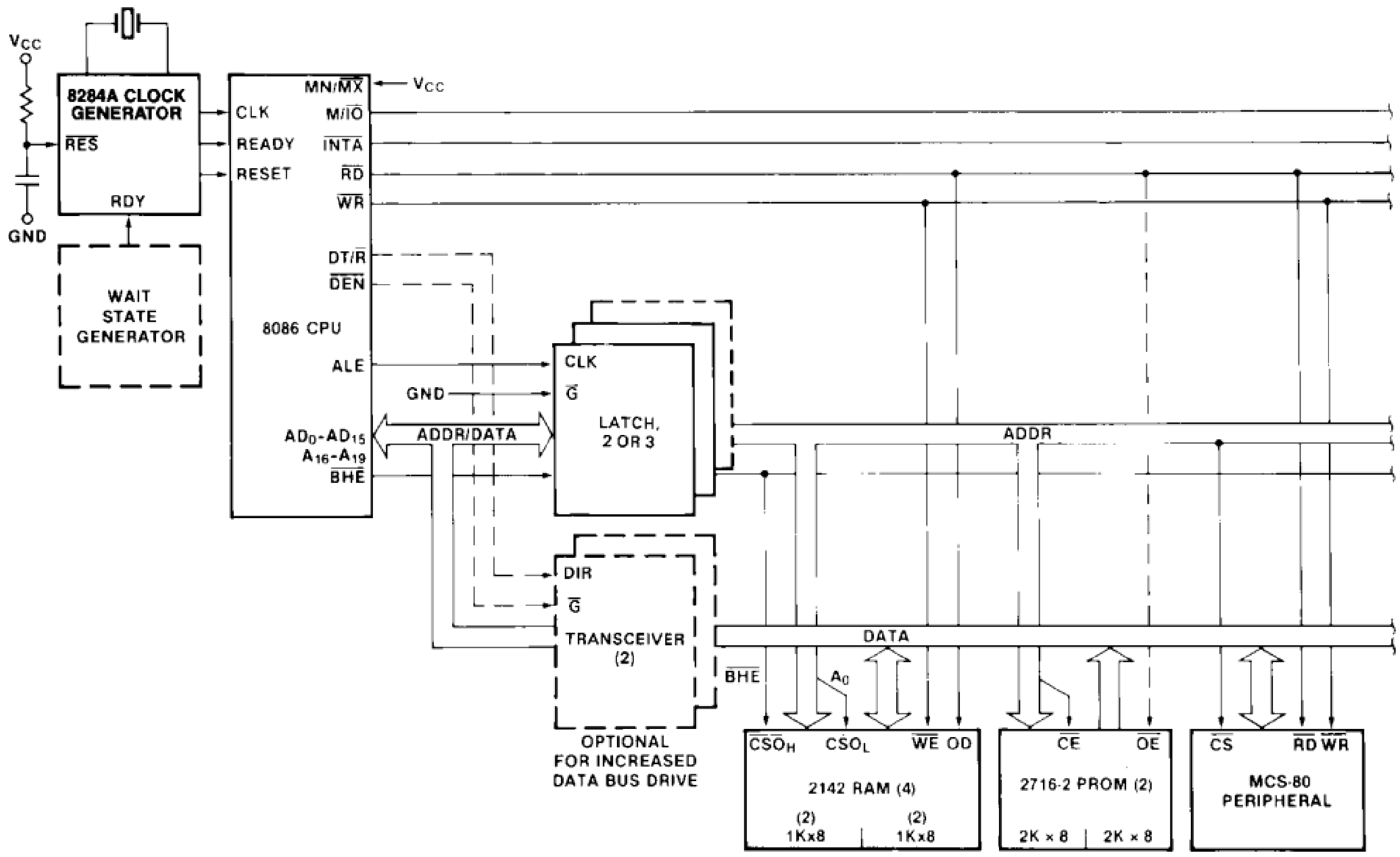
- Machine cycles
- Microsystem design
  - Clock generator
  - Memory connection
  - Data transfer
- Main board architecture

Summary

- Single system bus



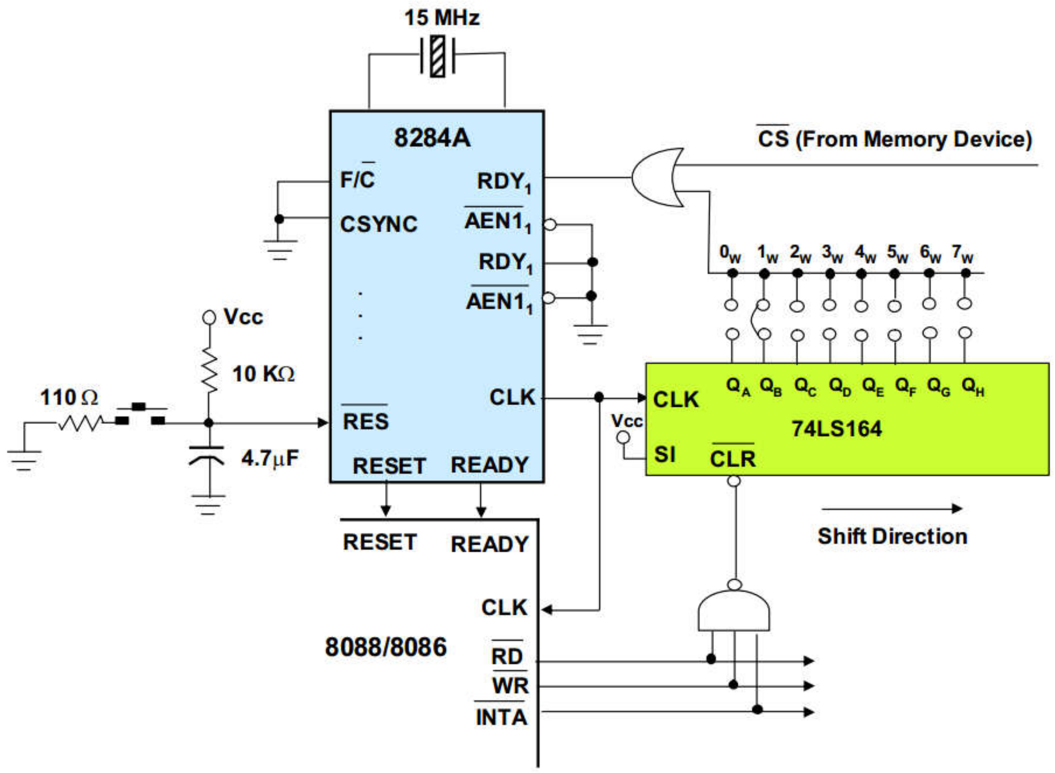
Summary



VCC
GND
8284A CLOCK GENERATOR
RES
RDY
WAIT STATE GENERATOR
CLK
READY
RESET
8086 CPU
MN/MX
M/IO
INTA
RD
WR
DT/R
DEN
ALE
AD0-AD15
A16-A19
BHE
GND
ADDR/DATA
CLK
G
LATCH, 2 OR 3
DIR
TRANSCEIVER (2)
OPTIONAL FOR INCREASED DATA BUS DRIVE
ADDR
DATA
BHE
A0
CSOH
CSOL
WE
OD
2142 RAM (4)
(2) 1Kx8
(2) 1Kx8
CE
OE
2716-2 PROM (2)
2K × 8
2K × 8
CS
RD
WR
MCS-80 PERIPHERAL

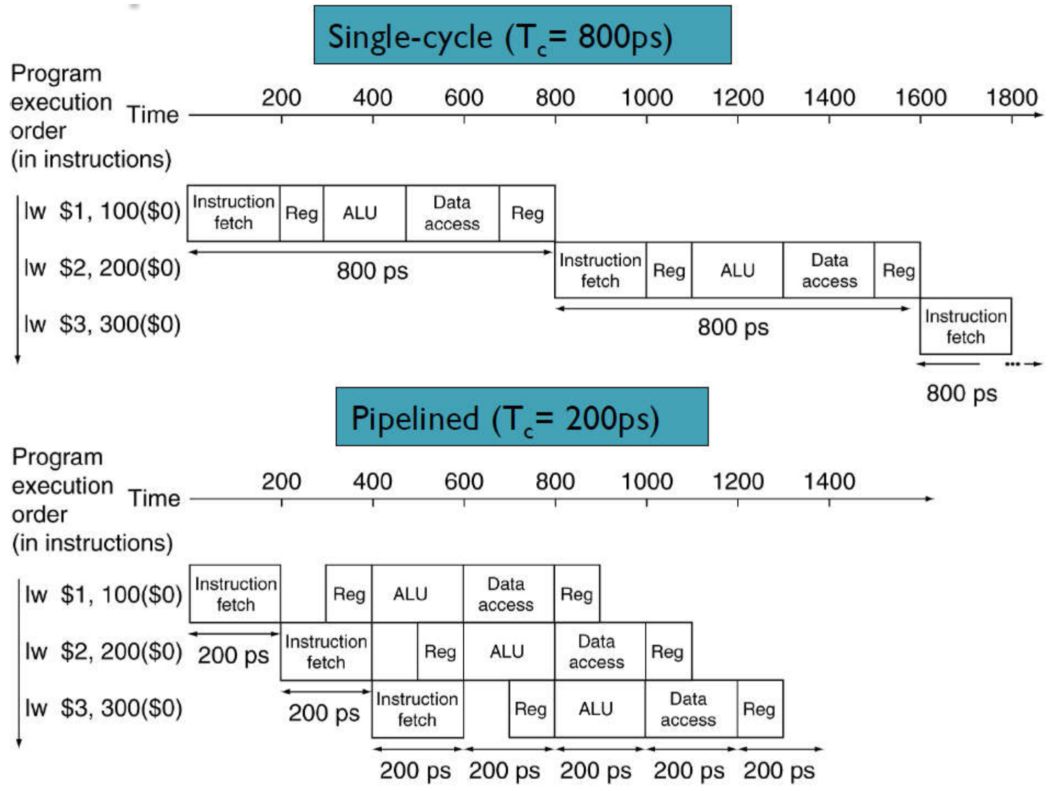
Machine cycles

- Wait states generator



Pipeline

- Pipelining speedup



Single-cycle (T_c= 800ps)
Program execution order (in instructions)
Time
200 400 600 800 1000 1200 1400 1600 1800
lw $1, 100($0)
lw $2, 200($0)
lw $3, 300($0)
Instruction fetch
Reg
ALU
Data access
Reg
800 ps
800 ps
800 ps
Pipelined (T_c= 200ps)
Program execution order (in instructions)
Time
200 400 600 800 1000 1200 1400
200 ps
200 ps
200 ps 200 ps 200 ps 200 ps 200 ps