

Lectures on

Vol 2

Computer Architecture & ASSEMBLY Programming

By

Dr Jeff Drobman

website → drjeffsoftware.com/classroom.html

email → jeffrey.drobman@csun.edu

Index

Vol 2

❖ CPU Org → slide 4

- ALU/FPU → 7
- Registers → 13
- Buses → 27

❖ Memory (RAM/ROM) → slide 36

- DRAM/SRAM → slide 44
- JEDEC → slide 68
- Endianness → slide 75
- Address Space → slide 81
- Segments → slide 84
- Cache → 90
- **Virtual Memory (MMU) → 107**

Course Description (122/222)

❖ Computer Architecture/Organization (COMP122/ 222)

☐ CPU, FPU, GPU org (ALU, registers, addressing)

☐ ISA's: MIPS, ARM, x86

☐ Memory models

- MLM- caches
- Virtual memory

☐ CPU status (PSW) & clock sync

☐ Interrupts, Exceptions, Syscall

☐ Cores & Threads

☐ Pipelines (ICU)

☐ Microprogramming (Am2900)

☐ Logic & State Machines (FSM)

☐ CPU performance/benchmarks

❖ Computer Arithmetic (COMP222)

☐ ALU: Full adder

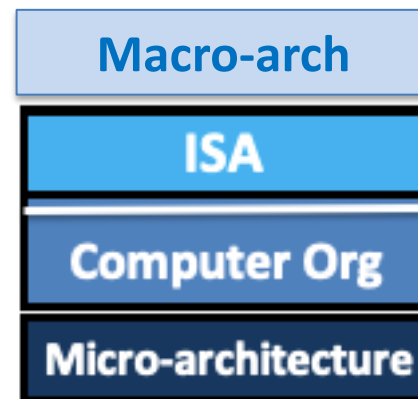
☐ Mult/Div (Booth's algorithm)

☐ Error codes (ECC, CRC, parity)

❖ Parallel & Micro Architecture (COMP222)

☐ Multi-core, Multi-threading, *superscalar*

☐ SIMD/MIMD/SPMD



System arch – Cores

Instructions (Primitives)

Software Interface

Execution Units

❖ ALU, ICU, Reg

Low-level execution

❖ Pipelines, threads

❖ scheduling

❖ branch prediction

(COMP122)

❖ Software Tools

☐ IDE's/Assemblers

☐ OS, RTOS, **Monitors**

☐ **Simulators**

❖ Debug Tools

☐ ICE/Logic Analyzers

☐ Disassemblers

Computer Architecture

CPU Organization

CPU Organization – ISA

COMP122

INSTRUCTION SET ARCHITECTURE

Memory – I/O

❖ Instruction Set

- ALU
- Branch/Jump (Loop)
- Call/Return (+RFI)
- Control bits/flags
- I/O ports
- Special (NOP)

❖ Memory Addressing Modes

- Direct (Immediate)
- Register Indirect (pointer)
- Base (reg) + offset (immed)

❖ REGISTER model

- General (GR)
- Dedicated

❖ MEMORY model

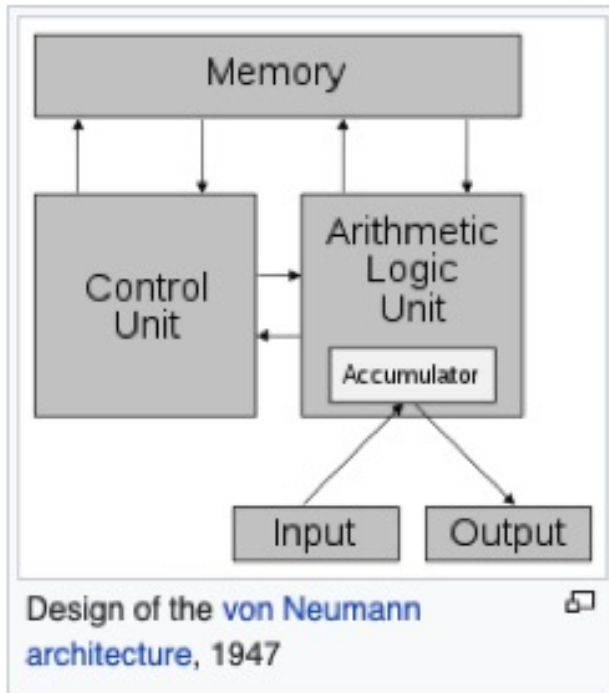
- Direct vs. Virtual (TLB)
- Segmented/paged
- Open vs. dedicated (reserved)
- Cache (I/D, L2/L3)

❖ I/O model

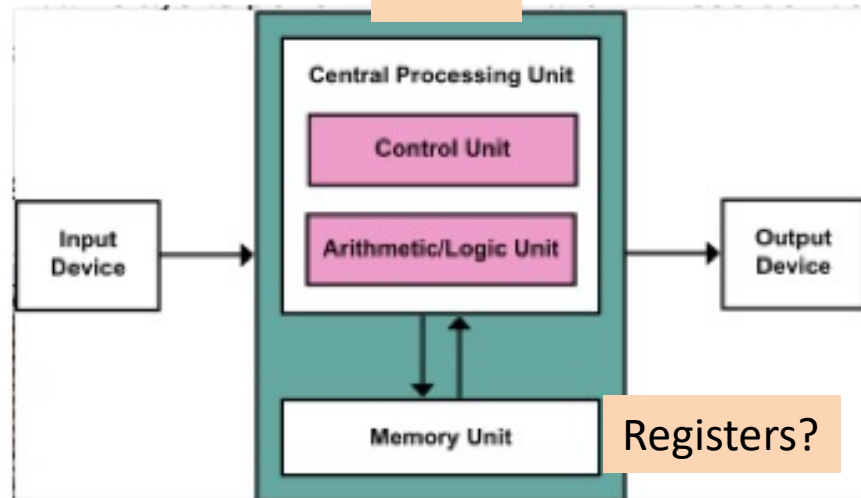
- Memory
 - Dedicated (IN/OUT)
 - **Memory mapped**
- Handling
 - Polling
 - **Interrupt**

1st Gen Architecture

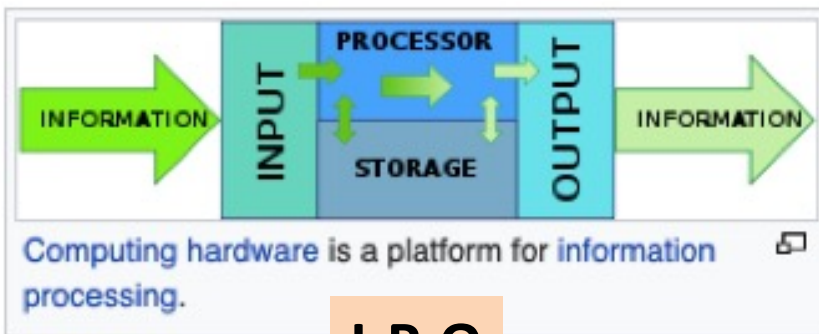
von Neumann



I-P-O

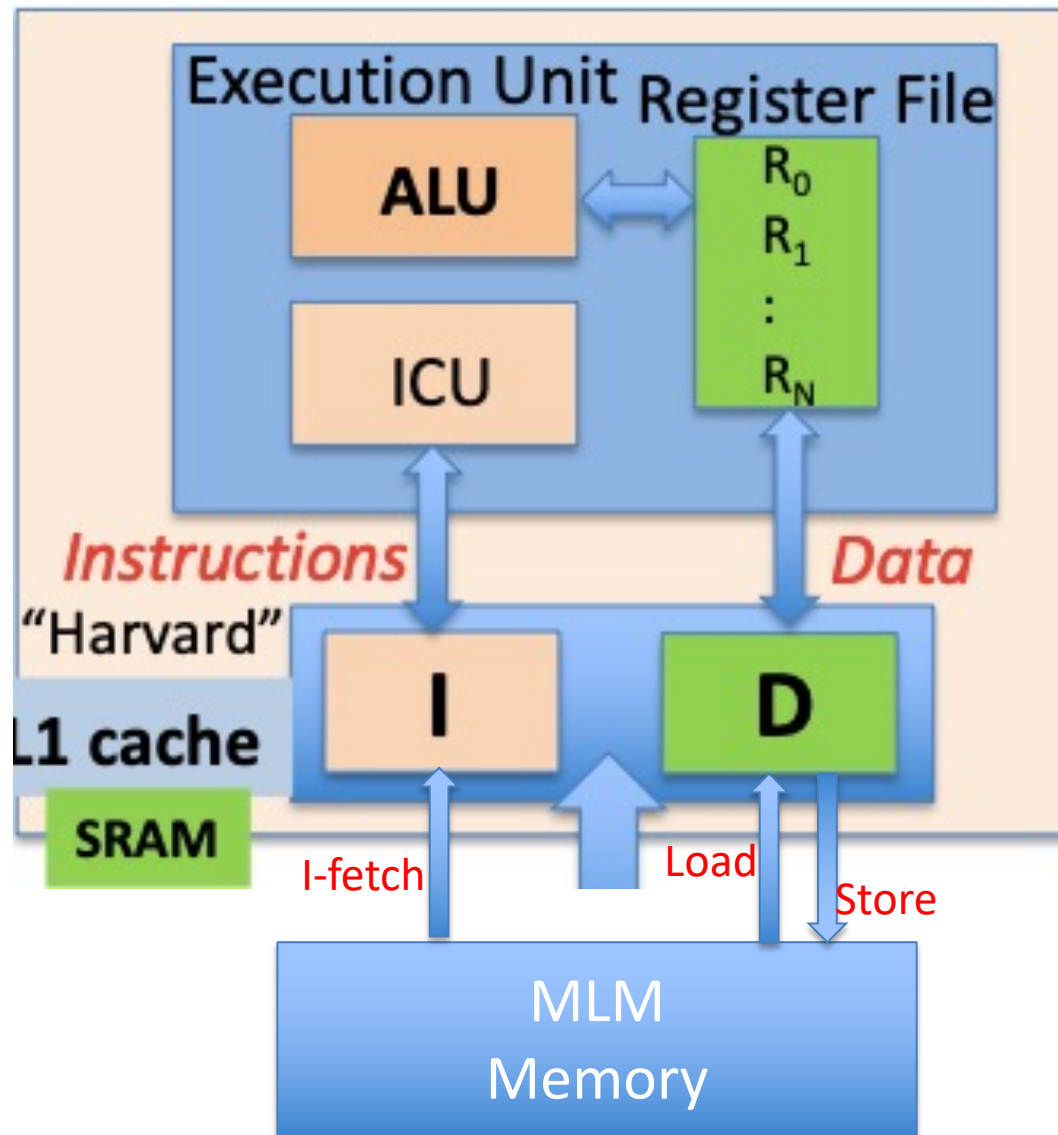


The **von Neumann architecture**—also known as the **von Neumann model** or **Princeton architecture**—is a computer architecture based on a 1945 description by Hungarian-American mathematician and physicist John von Neumann and others in the *First Draft of a Report on the EDVAC*. Th



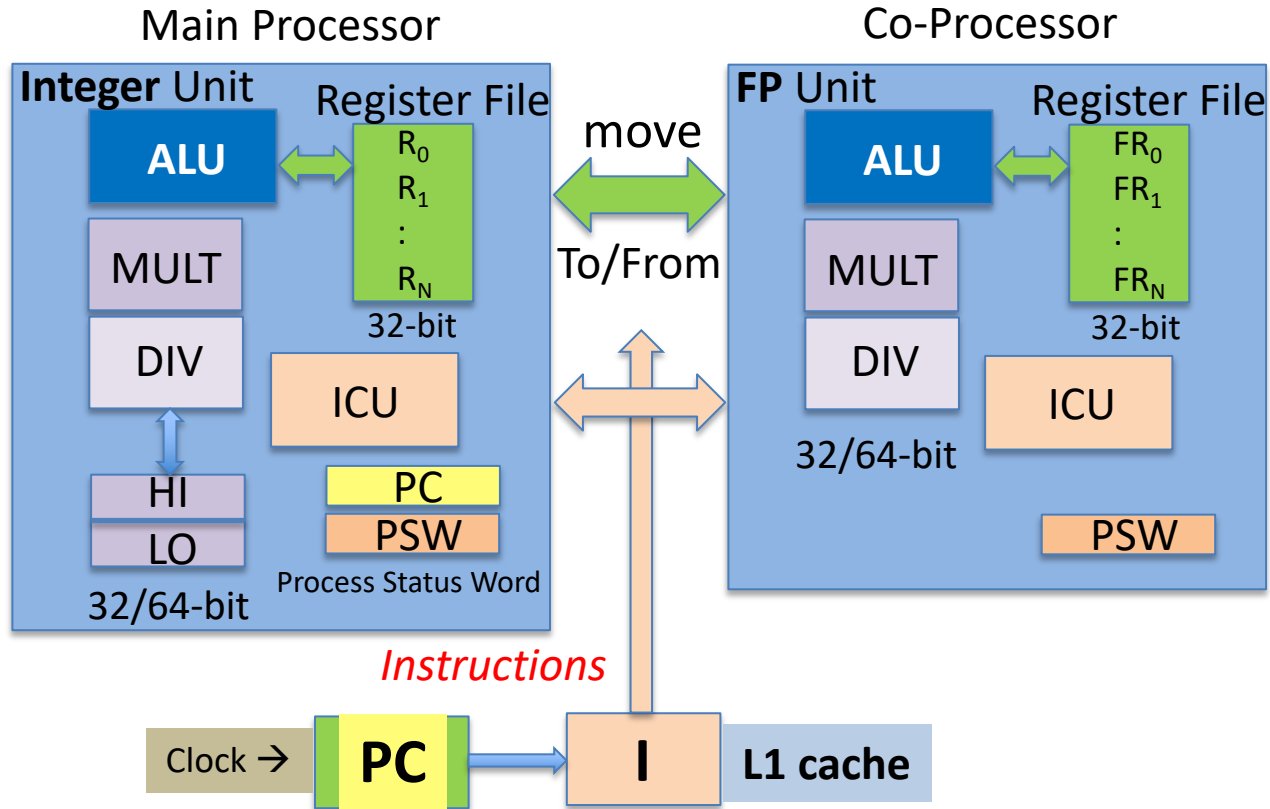
I-P-O

Integer Execution Unit (EU)



Integer Unit + FPU

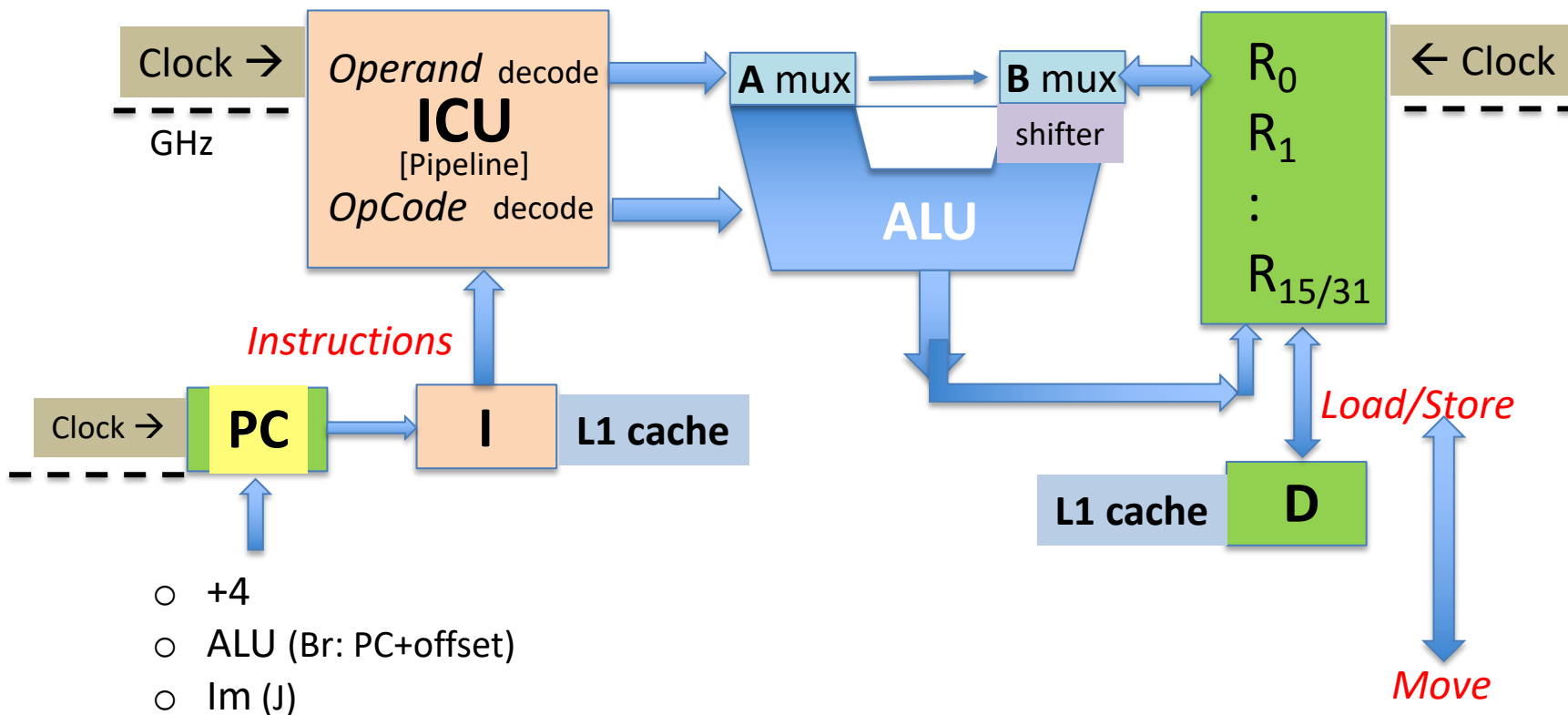
CPU = Integer Unit + FPU



Complete CPU Org

CPU: Integer *Execution Unit*

Clock → *synchronized*

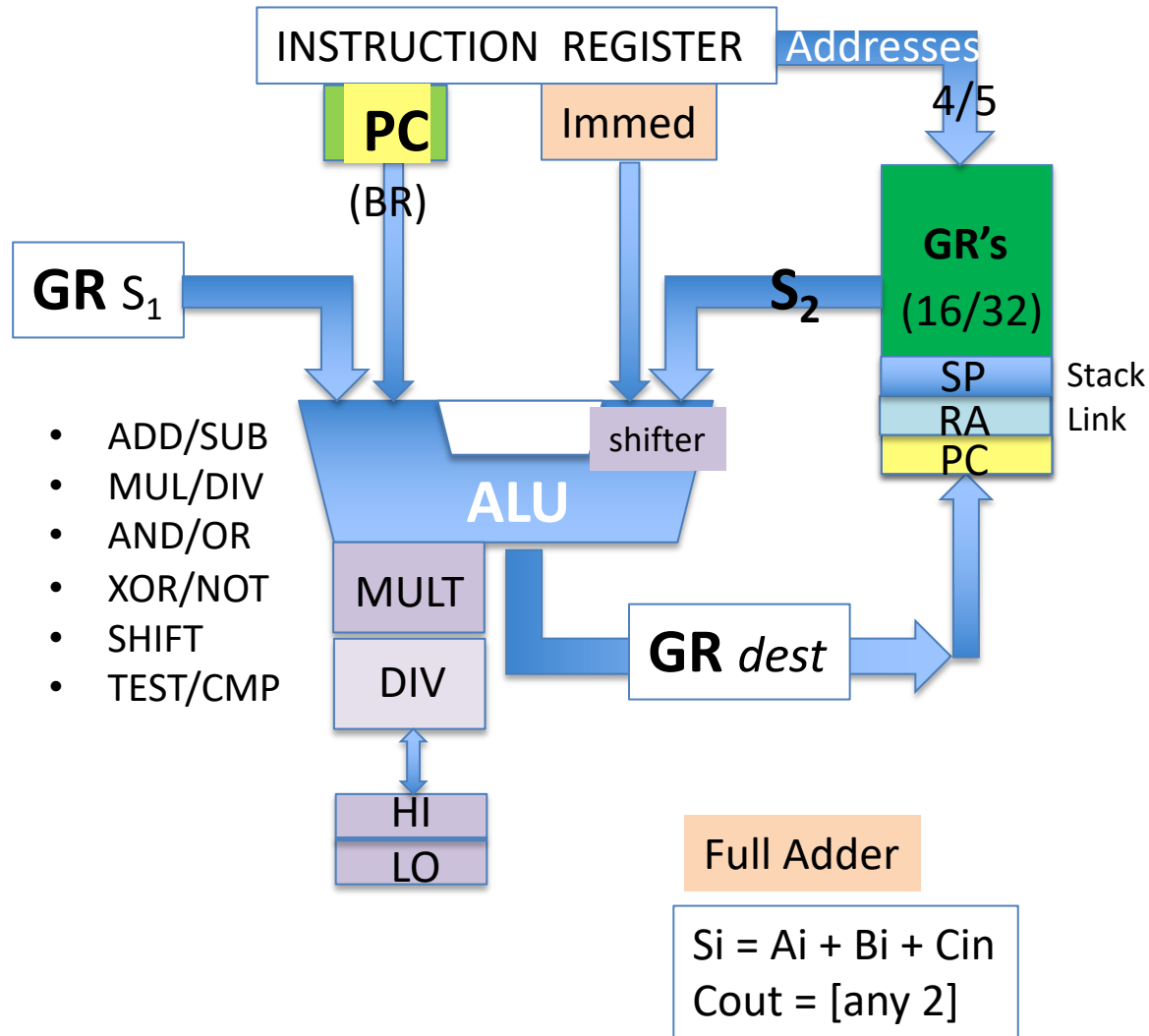


- +4
- ALU (Br: PC+offset)
- Im (J)

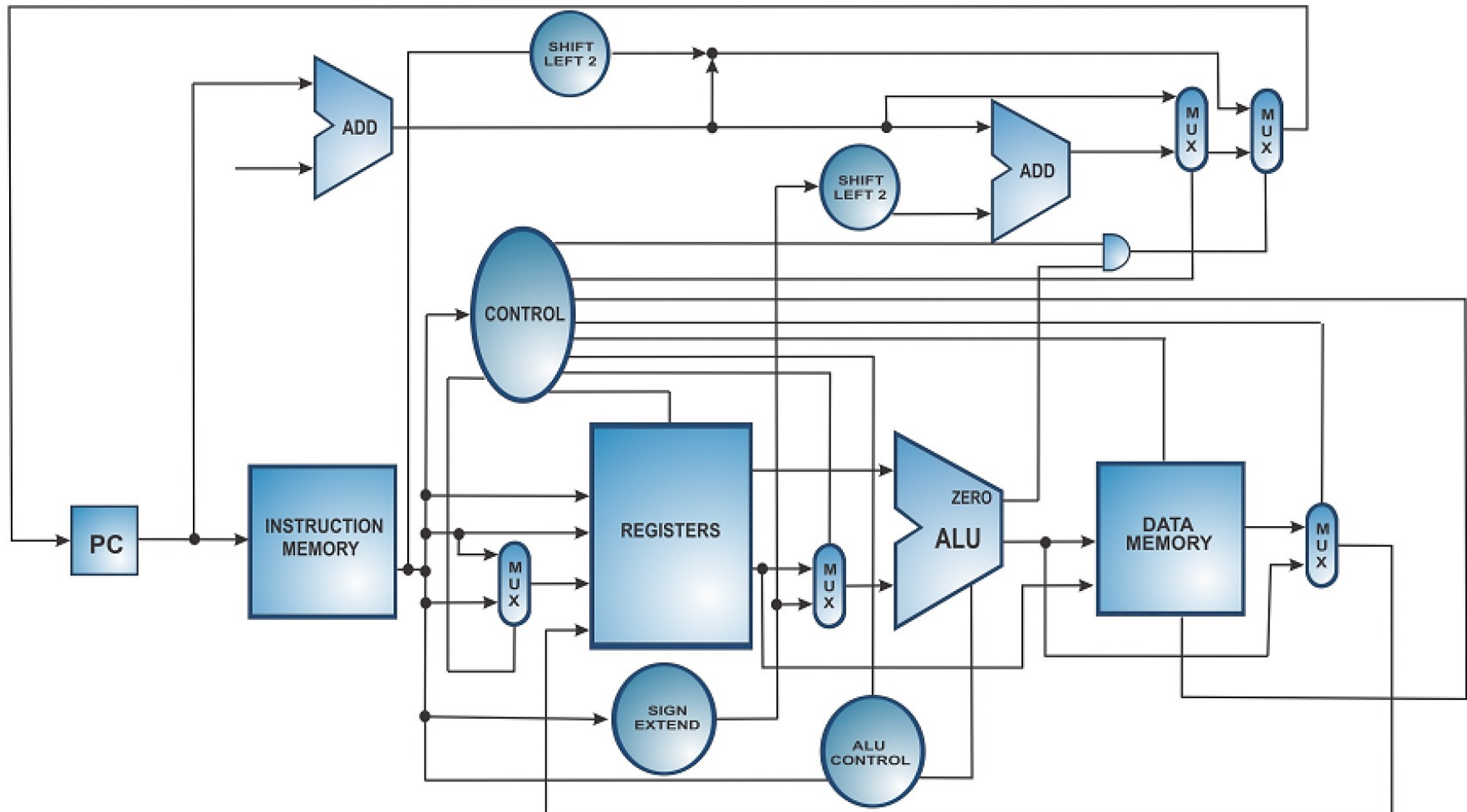
❖ FPU

- ☐ ALU
- ☐ Mult/Div/Sqrt
- ☐ FR's (32)

CPU: ALU Ops/Operands



MIPS MARS Xray



64-Bit CPU

What makes a computer 64-bit? Is it the OS version that you install? Does the RAM have to be more than 4GB? Does it have something to do with CPU architecture?



Jeff Drobman, Lecturer at California State University, Northridge (2016-present)

Answered just now

64-bit microprocessors have 64-bit data and addresses, but 32-bit instructions. It is the 64-bit addresses that is significant, as it extends memory address space from 4GiB (4 e9) to 16EiB (1.6 e19). That allows large OS's like Windows, MacOS and Linux more memory to use. So 64-bit OS's rely on this extra memory, use new 64-bit ISA's like ARMv8 or x86-64, and thus may not be able run most older 32-bit applications (without them being recompiled for the 64-bit ISA).

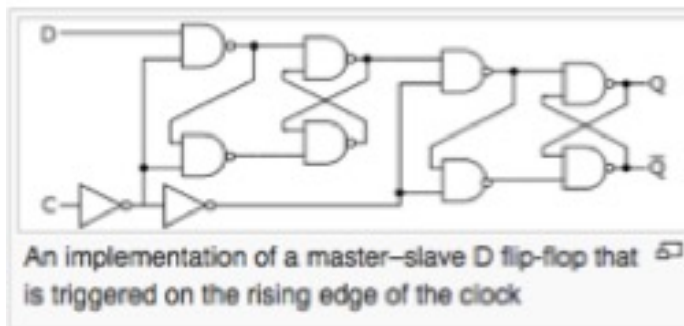
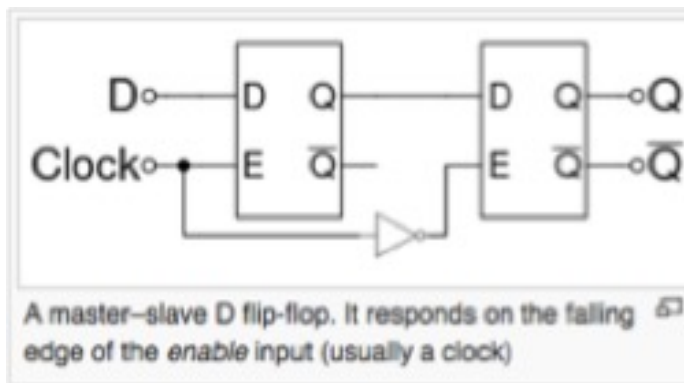
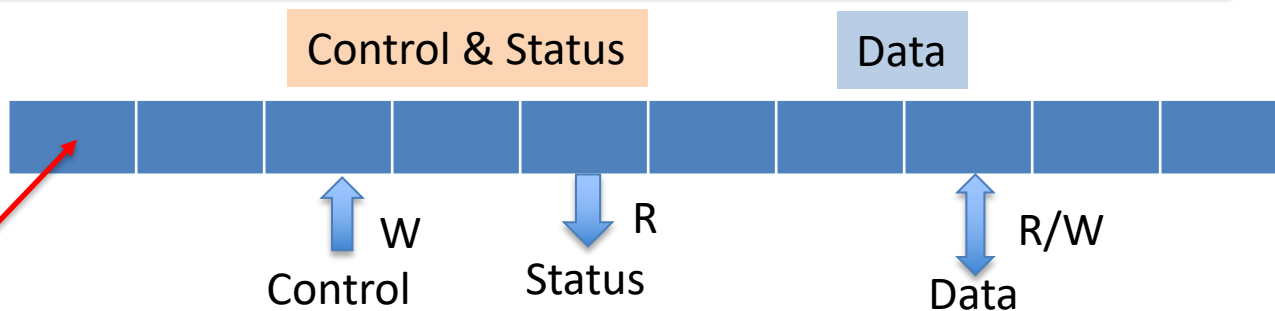
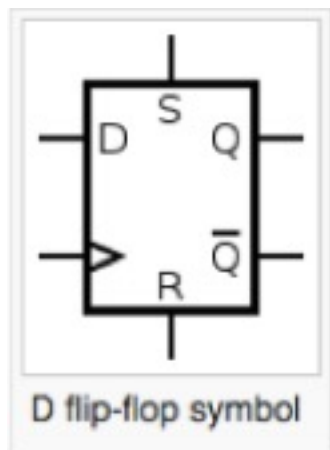
integer	2^{32}	4B	4.3×10^9
long	2^{64}	16 Q	1.84×10^{19}

16 EiB=18.4 EB

Computer Architecture

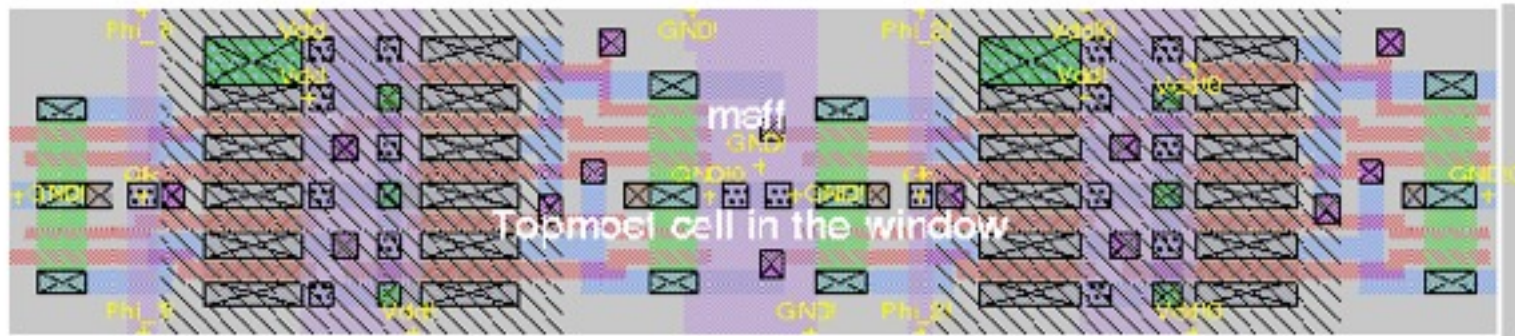
Registers

Registers



D Flip-Flops

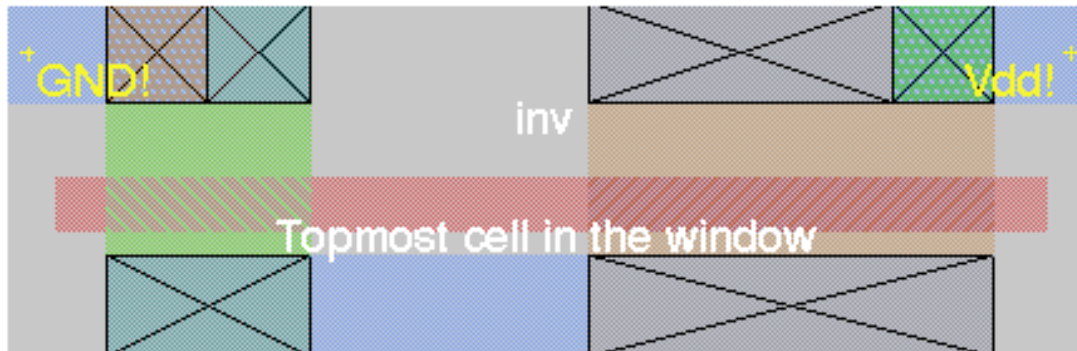
And if you want something more substantial to look at, here's one of my D-flip-flops, built out of clocked set-reset flip-flops in a master-slave configuration. A quick back-of-the-napkin estimate puts its area around $10000\mu m^2 = 0.01mm^2$, and if I counted correctly, there's 36 transistors.



I had a lot of fun laying that out. :-) If I could make the entire design this dense (ie. no wires), you could fit around 11,000 transistors on the die, and more if you use minimum-sized transistors. Realistically, wires take a ton of space, and you definitely don't want all your transistors to be minimum-sized. I had some largish transistors in my clock tree, for example.

D Flip-Flops: Inverter

For example, here's one of my inverters, with a $8\lambda \times 2\lambda$ N transistor on the left, and a $16\lambda \times 2\lambda$ P transistor on the right.



Key:

- The green area is N diffusion.
- The brown area is P diffusion.
- The reddish color is polysilicon. Where polysilicon crosses diffusion, you get a transistor.
- The light blue is Metal 1.
- The purplish color is Metal 2. (Not seen in the inverter, but present in the whole-chip picture above.)
- The boxes with Xs in them are contact points between layers, aka. *vias*.

Register Arch/Org

Hennessy & Patterson

Figure 2.21.1: The number of general-purpose registers in popular architectures over the years (COD Figure e2.21.1).

Machine	Number of general-purpose registers	Architectural style	Year
EDSAC	1	Accumulator	1949
IBM 701	1	Accumulator	1953
CDC 6600	8	Load-store	1963
IBM 360	16	Register-memory	1964
DEC PDP-8	1	Accumulator	1965
DEC PDP-11	8	Register-memory	1970
Intel 8008	1	Accumulator	1972
Motorola 6800	2	Accumulator	1974
DEC VAX	16	Register-memory, memory-memory	1977
Intel 8086	1	Extended accumulator	1978
Motorola 68000	16	Register-memory	1980
Intel 80386	8	Register-memory	1985
ARM	16	Load-store	1985
MIPS	32	Load-store	1985
HP PA-RISC	32	Load-store	1986
SPARC	32	Load-store	1987
PowerPC	32	Load-store	1992
DEC Alpha	32	Load-store	1992
HP/Intel IA-64	128	Load-store	2001
AMD64 (EMT64)	16	Register-memory	2003

CISC

RISC

16-bit MPU's



Advanced Micro Devices

CPU ISA's

❖ **MIPS**
❖ **ARM**

16-32 **GR's**

- ❖ Includes specials
 - ❑ Stack: SP, FP
 - ❑ Globals: GP
 - ❑ Zero
- ❖ 64-bit
 - ❑ *Paired* 32-bit

➤ **PC dedicated**

❖ **x86**

- ❑ Intel
- ❑ AMD

Dedicated Registers

- ❖ 4 Accumulators
 - ❑ A, B, C, D
- ❖ 4 Pointers
 - ❑ SI, DI
 - ❑ BP, SP
- ❖ 5 Memory Segments
 - ❑ CS, DS, ES, FS, GS, SS
- ❖ 64-bit
 - ❑ *Telescoping*

GP Registers

Register use convention:

Hennessy & Patterson

Figure 2.8.1: What is and what is not preserved across a procedure call (COD Figure 2.11).

If the software relies on the frame pointer register or on the global pointer register, discussed in the following subsections, they are preserved.

Preserved	Not preserved
Saved registers: \$s0–\$s7	Temporary registers: \$t0–\$t9
Stack pointer register: \$sp	Argument registers: \$a0–\$a3
Return address register: \$ra	Return value registers: \$v0–\$v1
Stack above the stack pointer	Stack below the stack pointer

- ❖ \$a(0:3) *args*
- ❖ \$at, \$k(0:1) *reserved*
- ❖ \$v(0:1) *values*
- ❖ \$t(0-9) *temp*
- ❖ \$s(0:7) *saved*
- ❖ \$gp *global ptr*
- ❖ \$sp *stack ptr*
- ❖ \$fp *frame ptr*
- ❖ \$ra *return addr*

MARS (MIPS Assembler and Runtime Simulator)

Registers

The screenshot shows the MARS application window with the 'Registers' panel selected for 'Coproc 0'. The panel displays a table of registers with their names, numbers, and values.

Name	Number	Value
\$8 (vaddr)	8	0x00000000
\$12 (status)	12	0x0000ff11
\$13 (cause)	13	0x00000000
\$14 (epc)	14	0x00000000

The screenshot shows the MARS application window with the 'Registers' panel selected for 'Coproc 0'. The panel displays a table of floating-point registers with their names and values.

Name	Float
\$f0	0x00000000
\$f1	0x00000000
\$f2	0x00000000
\$f3	0x00000000
\$f4	0x00000000
\$f5	0x00000000
\$f6	0x00000000
\$f7	0x00000000
\$f8	0x00000000
\$f9	0x00000000
\$f10	0x00000000

The screenshot shows the MARS application window with the 'Registers' panel selected for 'Coproc 0'. The panel displays a table of integer registers with their names, numbers, and values.

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x00000000
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000
\$t1	9	0x00000000
\$t2	10	0x00000000
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x00000000
\$s1	17	0x00000000
\$s2	18	0x00000000
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$t8	24	0x00000000
\$t9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10008000
\$sp	29	0x7ffffc
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00400000
hi		0x00000000
lo		0x00000000

ARM(v5/7) Regs – ARMSim

ARMv5 → 16 GR's

ARMv7 → 32 GR's

```
R0      : 000010a0
R1      : 00000001
R2      : 65707954
R3      : 00004014
R4      : 00000037
R5      : 00000002
R6      : 00000000
R7      : 00000000
R8      : 00000000
R9      : 00000000
R10 (s1) : 00000000
R11 (fp) : 00000000
R12 (ip) : 00000000
R13 (sp) : 00011400
R14 (lr) : 00001024
R15 (pc) : 00001058
```

CPSR Register

Negative (N) : 0

Zero (Z) : 1

Carry (C) : 1

Overflow (V) : 0

IRQ Disable: 1

FIQ Disable: 1

Thumb (T) : 0

CPU Mode : System

0x600000df

x86 Registers

Registers [\[edit \]](#)

Further information: [X86 architecture](#) § [x86 registers](#)

x86 processors have a collection of registers available to be used as stores for binary data. Collectively each register has a special purpose in addition to what they can all do:

- AX multiply/divide, string load & store
- CX count for string operations & shifts
- DX [port](#) address for IN and OUT
- BX index register for MOVE
- SP points to top of [the stack](#)
- BP points to base of the stack frame
- SI points to a source in stream operations
- DI points to a destination in stream operations

Along with the general registers there are additionally the:

- IP instruction pointer
- [FLAGS](#)
- segment registers (CS, DS, ES, FS, GS, SS) which determine where a 64k segment starts (no FS)
- extra extension registers ([MMX](#), [3DNow!](#), [SSE](#), etc.) (Pentium & later only).

The IP register points to the memory offset of the next instruction in the code segment (it points to the by the programmer directly).

The x86 registers can be used by using the [MOV](#) instructions. For example, in Intel syntax:

```
mov ax, 1234h ; copies the value 1234hex (4660d) into register AX
```

```
mov bx, ax    ; copies the value of the AX register into the BX register
```

x86 Registers

16-bit 20-bit address

Intel 8086 registers

1 9 1 8 1 7 1 6 1 5 1 4 1 3 1 2 1 1 0 9 8 7 6 5 4 3 2 1 0 (bit position)

Main registers

	AH	AL	AX (primary accumulator)
	BH	BL	BX (base, accumulator)
	CH	CL	CX (counter, accumulator)
	DH	DL	DX (accumulator, extended acc.)

Index registers

0 0 0 0	SI	Source Index
0 0 0 0	DI	Destination Index
0 0 0 0	BP	Base Pointer
0 0 0 0	SP	Stack Pointer

Program counter

0 0 0 0	IP	Instruction Pointer
---------	-----------	---------------------

Segment registers

CS	0 0 0 0	Code Segment
DS	0 0 0 0	Data Segment
ES	0 0 0 0	Extra Segment
SS	0 0 0 0	Stack Segment

Status register

- - - - **O D I T S Z** - **A** - **P** - **C** Flags

x86 Registers

32-bit

Hennessy & Patterson

Figure 2.17.1: The 80386 register set (COD Figure 2.36).

Starting with the 80386, the top eight registers were extended to 32 bits and could also be used as general-purpose registers.

Name	31	0	Use
EAX			GPR 0
ECX			GPR 1
EDX			GPR 2
EBX			GPR 3
ESP			GPR 4
EBP			GPR 5
ESI			GPR 6
EDI			GPR 7
	CS		Code segment pointer
	SS		Stack segment pointer (top of stack)
	DS		Data segment pointer 0
	ES		Data segment pointer 1
	FS		Data segment pointer 2
	GS		Data segment pointer 3
EIP			Instruction pointer (PC)
EFLAGS			Condition codes

x86 Registers

64-bit

Structure [[edit](#)]

General Purpose Registers (A, B, C and D)

64	56	48	40	32	24	16	8
R?X							
				E?X			
						?X	
						?H	?L

64-bit mode-only General Purpose Registers (R8, R9, R10, R11, R12, R13, R14, R15)

64	56	48	40	32	24	16	8
?							
				?D			
						?W	
						?B	

Segment Registers (C, D, S, E, F and G)

16	8
?S	

Pointer Registers (S and B)

64	56	48	40	32	24	16	8
R?P							
				E?P			
						?P	
						?PL	

Note: The ?PL registers are only available in 64-bit mode.

Computer Architecture

Buses

Hardware-System Model

DROBMAN MODEL

- PRIMARY FUNCTIONS ARE NODES
- NODES ALSO CONTAIN THE OTHER TWO SUBORDINATE FUNCTIONS
- NODES ARE HIERARCHICAL
- INTERCONNECTIONS ARE GENERIC & BIDIRECTIONAL
- NODES MAY BE ROTATED FOR EMPHASIS (PRIMARY SYSTEM)

Primary Design Level:
Logic- Memory Cell

**MEMORY/
STORAGE**

transFIXES

COMMUNICATIONS

C

COMMUNICATIONS SYSTEM

M

**MEMORY/
STORAGE**

P

PROCESSOR
(E.G., NETWORK PROCESSOR)

TRIPARTITE GRAPH MODEL OF DIGITAL SYSTEMS: **COMPUTER**

P

PROCESSOR

transPOSES/
transFORMS

Primary Design Level:
Logic

LOAD - STORE

IN - OUT

Primary Design Level:
Physical

**INPUT/OUTPUT
OR
COMMUNICATIONS**

(INCLUDES PERIPHERALS)

DMA

transFERS/
transPORTS

I/O

M

STORAGE SYSTEM

M

**MEMORY/
STORAGE**

P

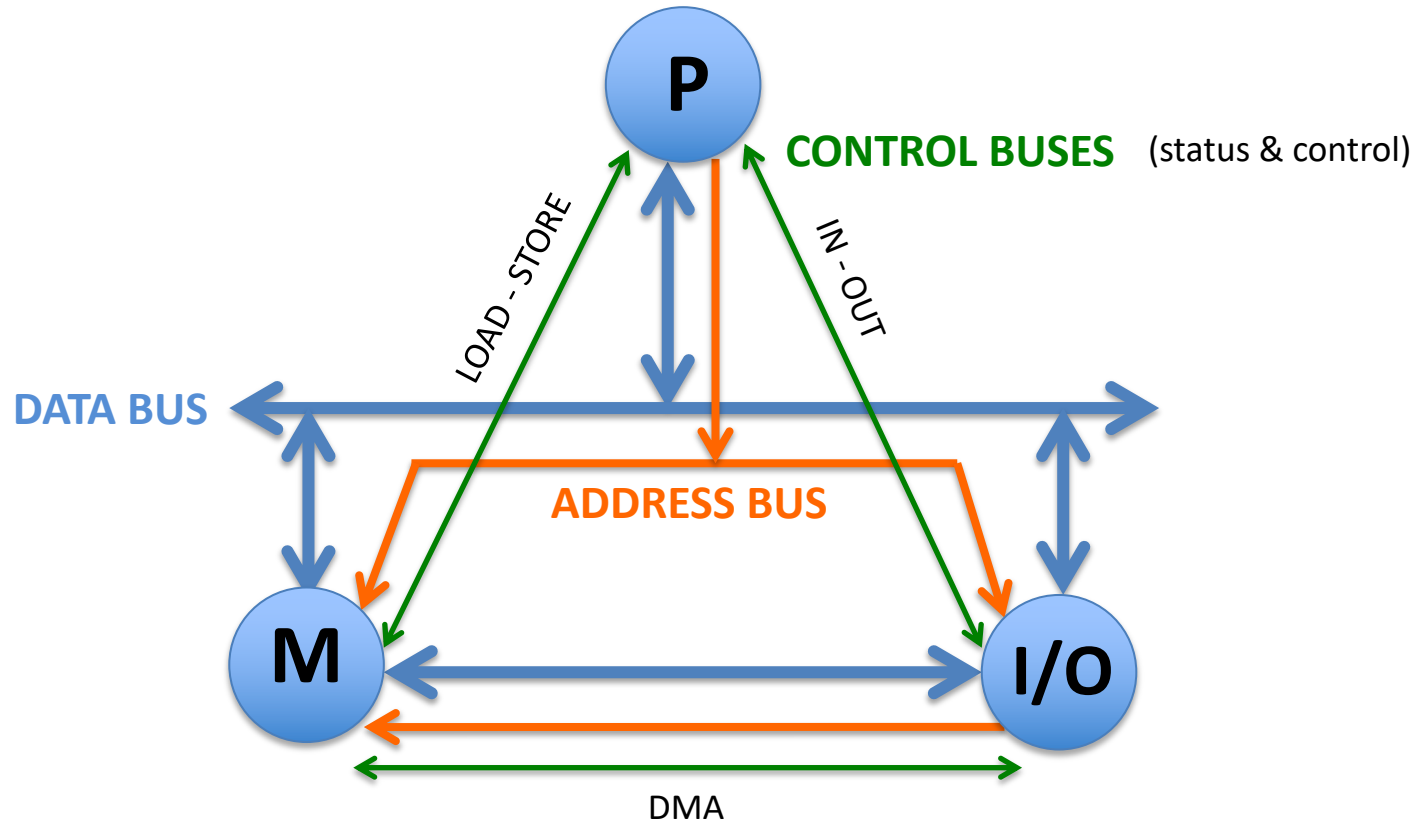
PROCESSOR

I/O

INPUT/OUTPUT

System Buses

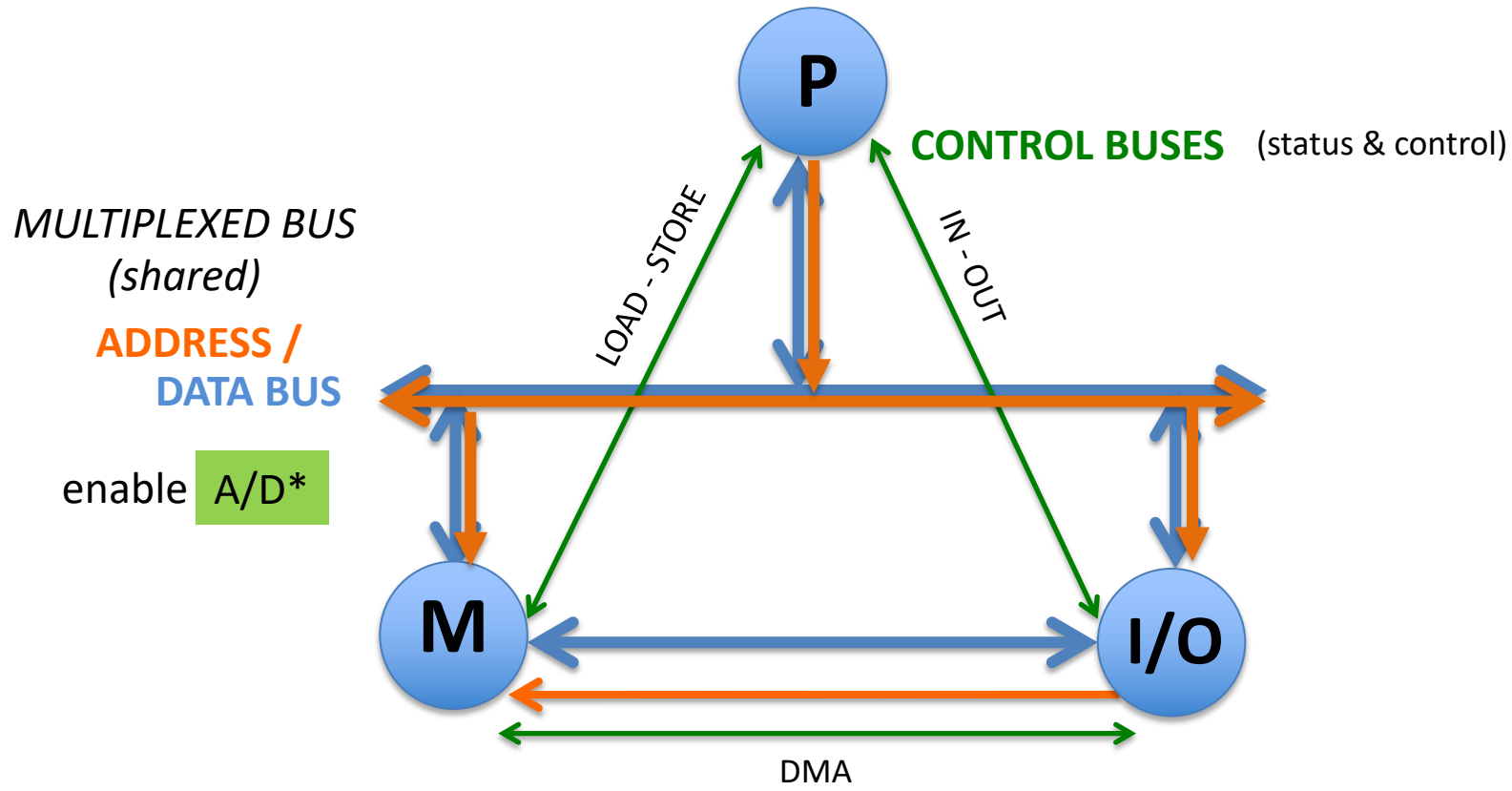
NON-Multiplexed A+D Buses



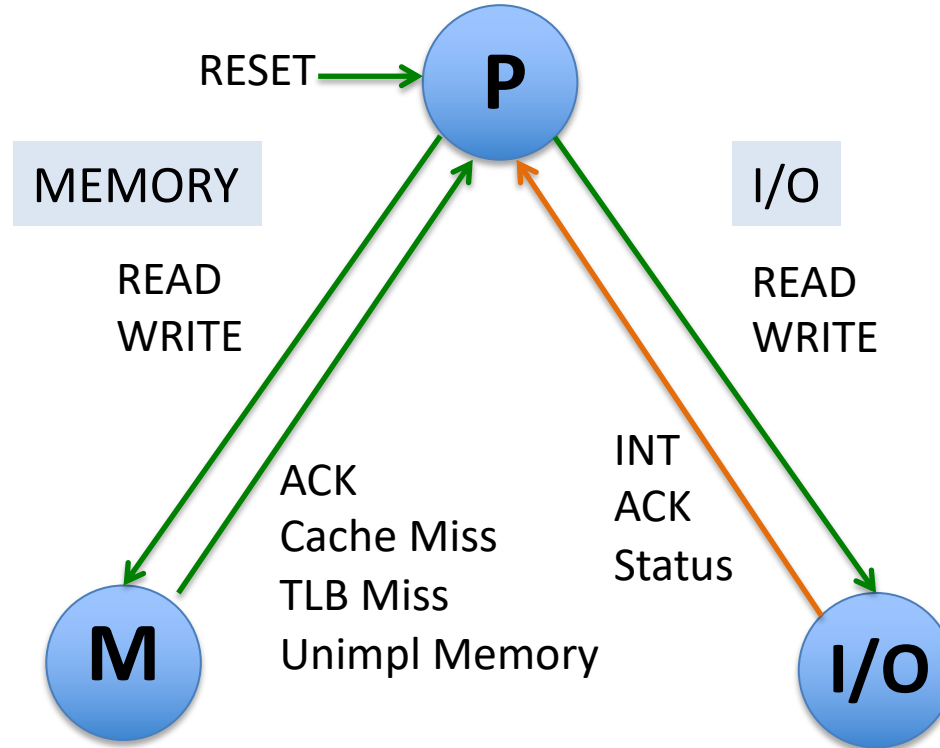
NON-MULTIPLEXED BUSES

System Buses

Multiplexed A/D Bus



Control Signals



- ✧ Intel mode $\rightarrow R^*, W^*$
- ✧ Motorola mode $\rightarrow R/W^*, DS^*$

- ✧ Polling
- ✧ Interrupts

Interrupts

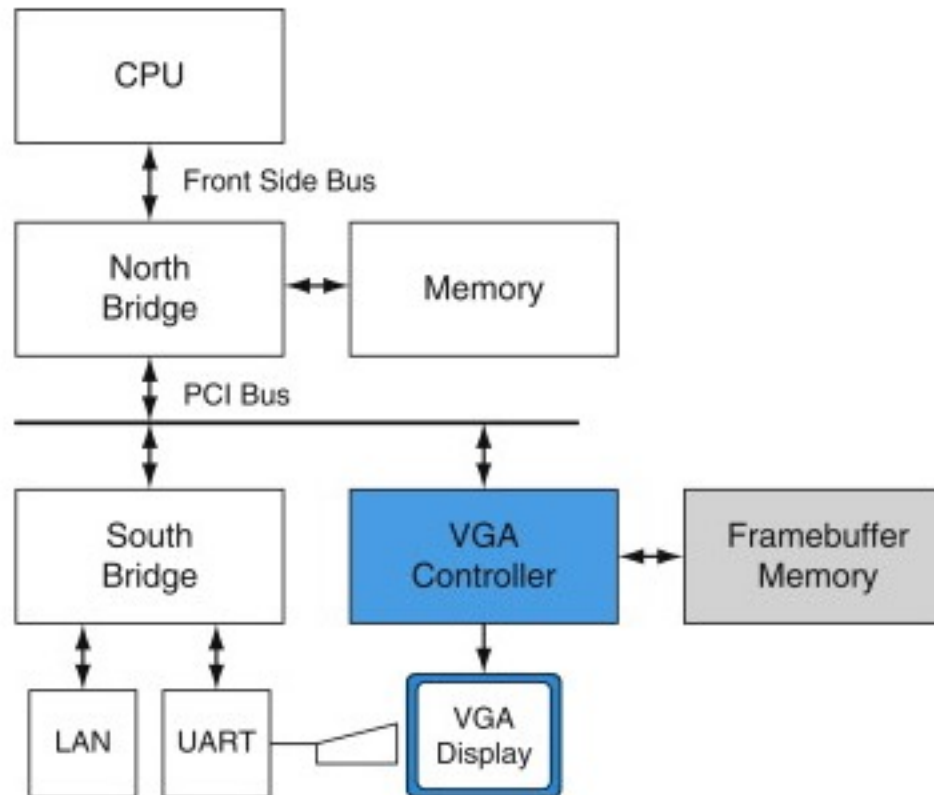
- NMI
- NVI
- VI

System PCI Bus & Bridges

Hennessy & Patterson

Figure 8.2.1: Historical PC (COD Figure B.2.1).

VGA controller drives graphics display from framebuffer memory.

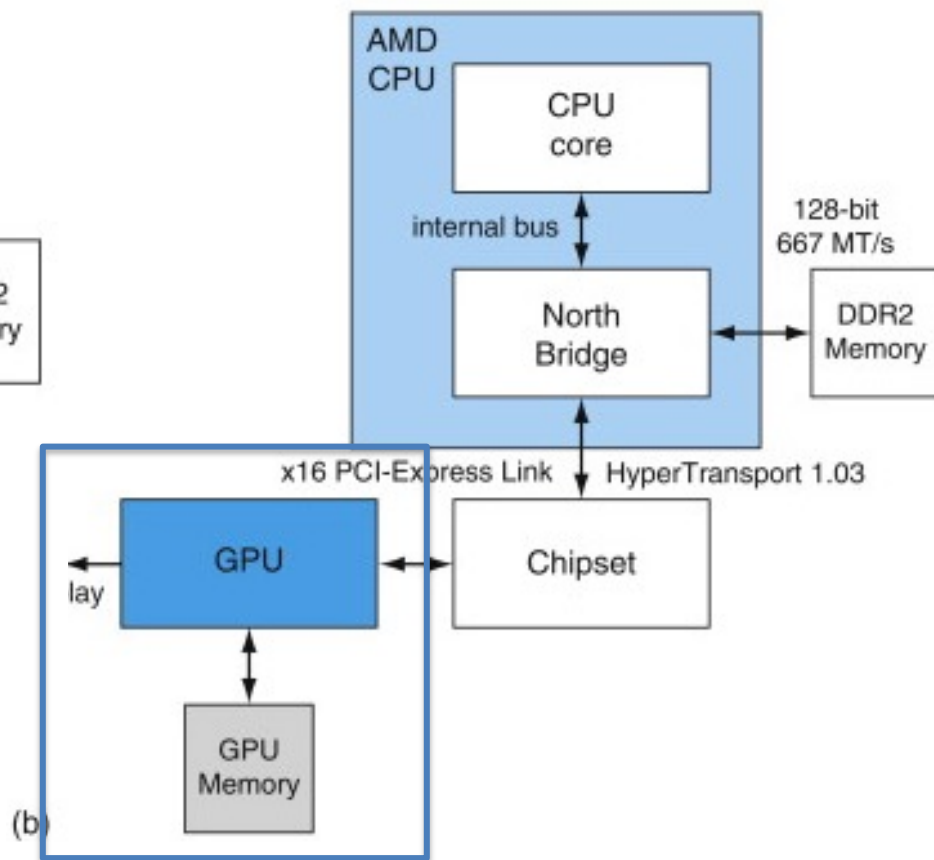
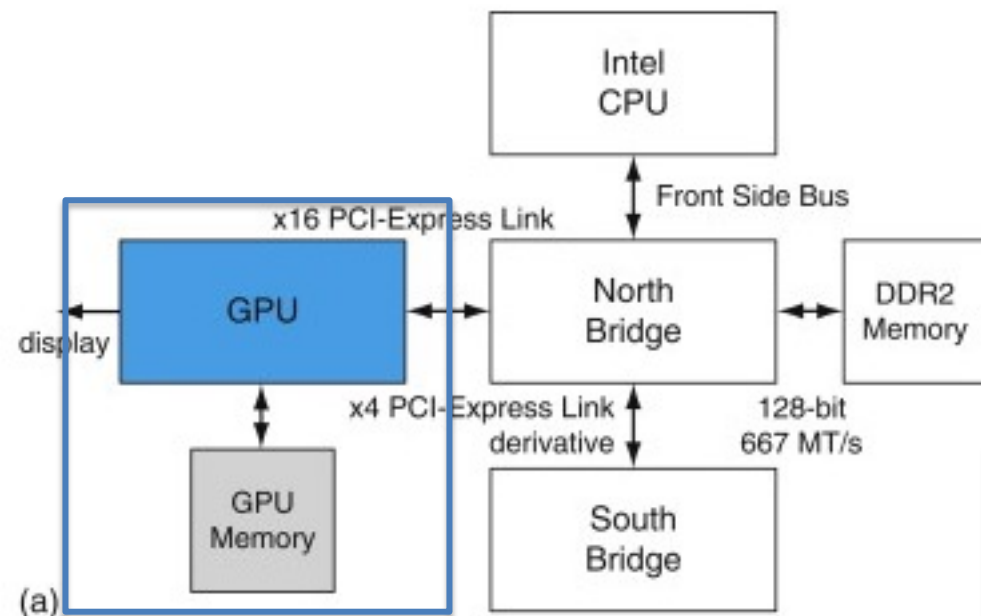


APU = CPU + GPU

Hennessy & Patterson

Intel

AMD



➤ See separate slide set "GPU"

Memory Buses

Are system buses and memory channels the same thing?



Andrew Silverman, Occupied in design/manufacturing of consumer electronics.



Answered 6h ago

In early computers they were often the same, but today they are generally separate, which allows the memory busses to run at potentially much higher overall data rates, and with different kinds of electrical signaling than peripheral devices on a different bus that don't require the same very high bandwidth. Additionally, having multiple memory channels avoids the bottleneck of all data to/from memory having to flow over a single electrical interface, increasing the amount of data that can be transferred simultaneously even more. In typical PCs today we usually see dual memory channels, and even more in high-end desktop or server architectures.

Memory Buses



Lawrence Stewart, Research Computer Scientist

Answered 6h ago



These terms belong to different eras of computer architecture, so it is difficult to compare, but they are not the same thing.

A long time ago, like 1970, some computers were built with a system bus, to which the CPU, memory, and I/O devices connected. A great example is the Unibus of the Digital Equipment PDP-11 minicomputers.

More recently, busses have fallen out of favor because they have real performance limitations. Instead, modern machines are built with point to point wires and switches. Instead of a bus, a machine will have an internal network to connect cores, caches, memory controllers, and I/O controllers. In this new environment, a memory channel is more or less equivalent to a memory controller. A computer may have many of them. A particular memory request will be routed by the network from a CPU core to a memory controller, and from there to the memory sticks or built-in memory or whatever.

So a memory channel is an independent memory controller linked to some memory, that can operate in parallel with other memory controllers.

Computer Architecture

Memory

Memory IC Timeline

RAM

1966

- ❖ **ROM**– 1st Semiconductor
- ❖ **DIP** packages
- ❖ **RAM**– Bipolar RAMs (SRAM) introduced
- ❖ **DRAM**– IBM conceives DRAM cell (1T, 1C)
- ❖ **CMOS SRAM**– 1st parts by RCA

1971

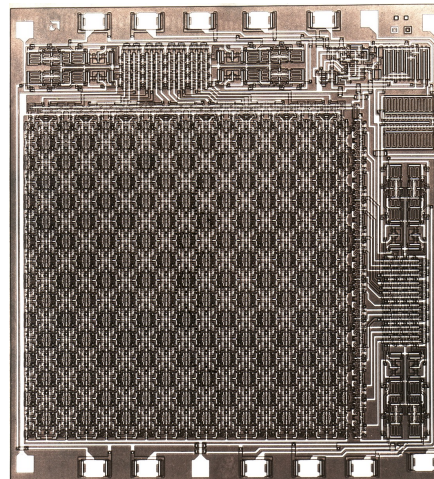
- ❖ **Microprocessor & RAM in MOS** invented by Intel

- ❖ i1101 256-bit **SRAM**
- ❖ i1103 1K-bit **DRAM**

(copied by AMD)

1987

- ❖ **Toshiba** intro's **Flash** EEPROM,



Am1101A die

Memory Types

Wiki

Computer memory types

Volatile

RAM

DRAM (SDRAM · DDR · GDDR · HBM) ·
SRAM

Historical

Williams–Kilburn tube (1946–47) ·
Delay line memory (1947) ·
Mellon optical memory (1951) · Selectron tube
(1952) · Dekatron · T-RAM (2009) · Z-RAM
(2002–2010)

Non-volatile

ROM

Mask ROM · PROM · EPROM · EEPROM ·

Flash memory

NVRAM

ReRAM

Early stage NVRAM

FeRAM · MRAM · PCM (3D XPoint) ·
FeFET memory

Magnetic

Magnetic tape data storage
(Linear Tape-Open) · Hard disk drive

Optical

Optical disc · 5D optical data storage

In development

CBRAM · Racetrack memory · NRAM ·
Millipede memory · ECRAM

Historical

Paper data storage (1725) · Drum memory
(1932) · Magnetic-core memory (1949) ·
Plated wire memory (1957) ·
Core rope memory (1960s) · Thin-film memory
(1962) · Disk pack (1962) · Twistor memory
(~1968) · Bubble memory (~1970) · Floppy disk
(1971)

Memory Speeds

COMP122



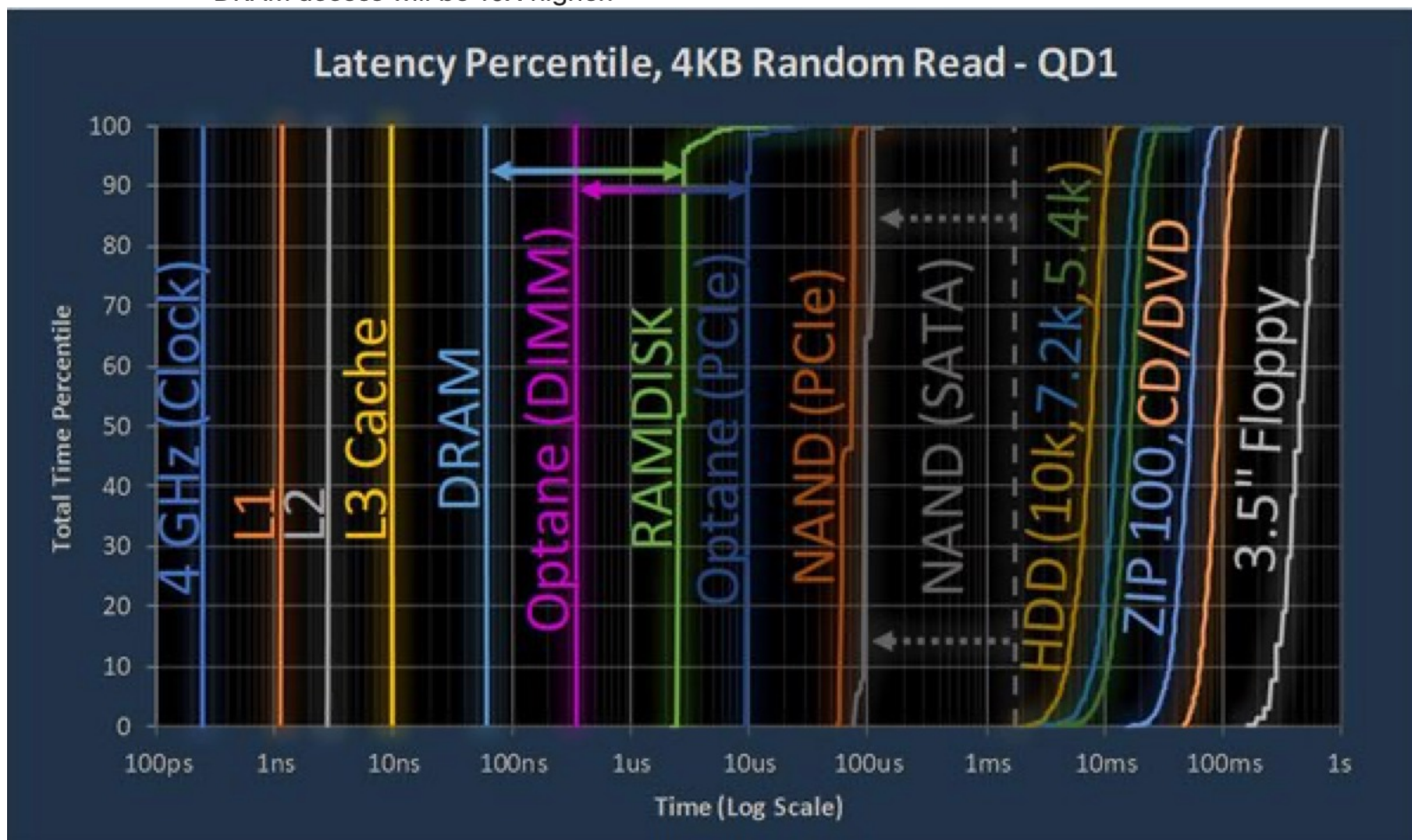
Yowan Rajcoomar, Computer Technician (2008-present)

Answered Nov 10



This will depend on the exact type of SRAM and the cell library/density used. The least dense SRAM levels (L1 caches) will have the lowest access times.

Generally speaking, the L1 SRAM cache will have access times of under 1ns while DRAM access will be 10X higher.



Memory Speeds: Core i9-12

Example with a Core i9-12900K showing access times of L1, L2 and L3 caches alongside DDR5 SDRAM:

AIDA64 Cache & Memory Benchmark

	Read	Write	Copy	Latency
Memory	81497 MB/s	74244 MB/s	73925 MB/s	77.1 ns
L1 Cache	2308.5 GB/s	1424.7 GB/s	4135.2 GB/s	1.0 ns
L2 Cache	1283.6 GB/s	532.90 GB/s	899.71 GB/s	2.9 ns
L3 Cache	900.71 GB/s	462.40 GB/s	737.60 GB/s	18.5 ns
CPU Type	8C+8c Intel Core i9-12900K (Alder Lake-S, LGA1700)			
CPU Stepping	C0/H0			
CPU Clock	4900.0 MHz			
CPU FSB	100.0 MHz (original: 100 MHz)			
CPU Multiplier	49x	North Bridge Clock		2200.0 MHz
Memory Bus	2600.0 MHz	DRAM:FSB Ratio		26:1
Memory Type	Quad Channel DDR5-5200 SDRAM (40-40-40-76 CR2)			
Chipset	Intel Alder Point-S Z690, Intel Alder Lake-S			
Motherboard	Asus ROG Maximus Z690 Hero			
BIOS Version	0604			

AIDA64 v6.33.5772 Beta / BenchDLL 4.5.859.8-x64 (c) 1995-2021 FinalWire Ltd.

ROM

What was the development that allowed computer chips to retain their memory without constant power being applied?



Jeff Drobman, Lecturer at California State University, Northridge (2016-present)

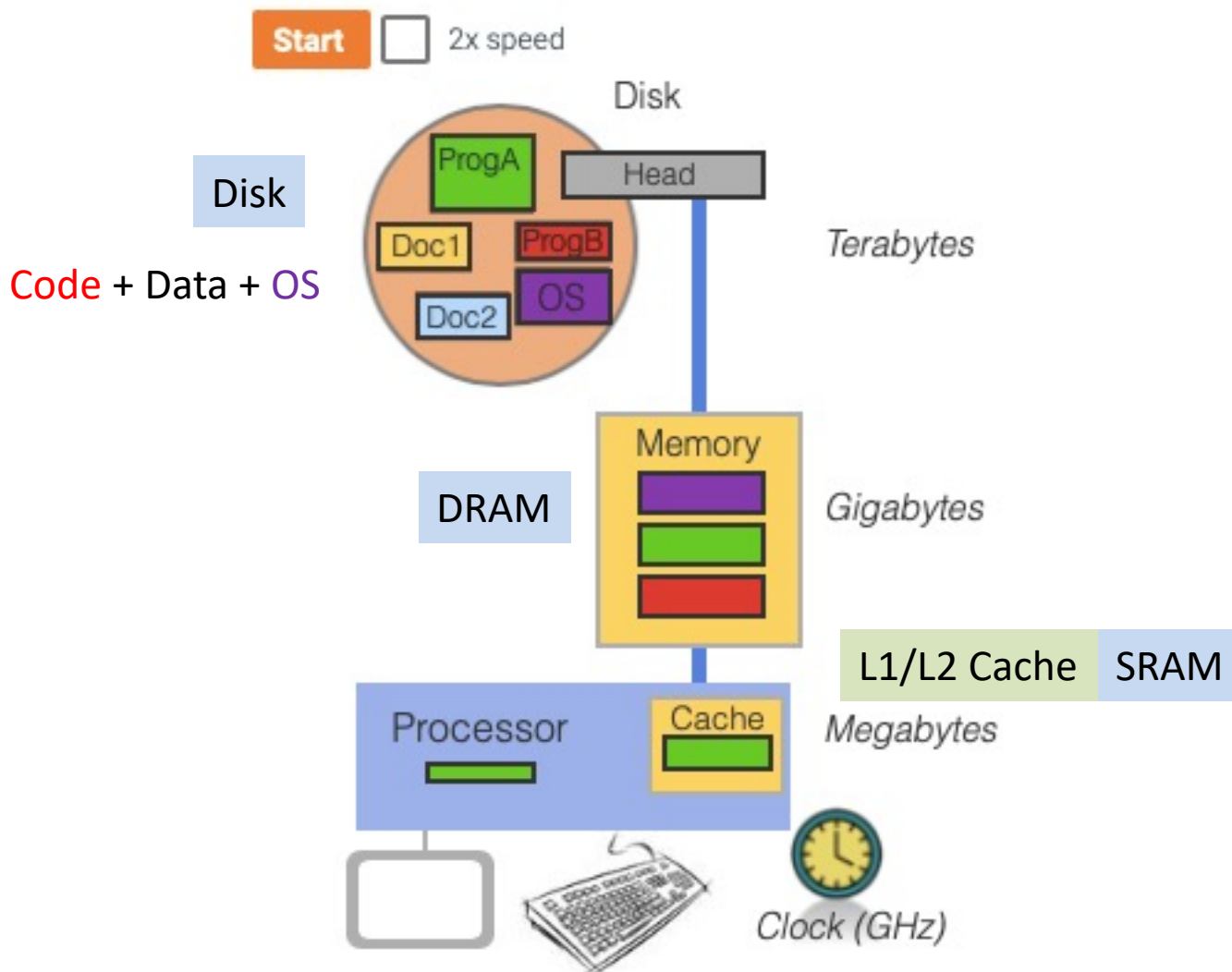
Answered just now

that would be a ROM. the 1st ones were mask made: using diodes and fuses per bit that were blown or "burned" in. next generation used "programmable" fuses: PROM. then erasable, so EPROM — via UV light to erase one word at a time. finally, we got EEPROM — electrically erasable, so in situ (on the PCB), and they became the first pseudo-writable ROM (very slow to write). that was fixed with "flash" erasable in blocks not words at a time, and ever faster to write. we call all the types of memory that saves data regardless of power "NVM" — non-volatile memory.

ROM → PROM → EPROM → EEPROM → Flash

Computer Memory Org

1.6.1: Some computer components.



Memory

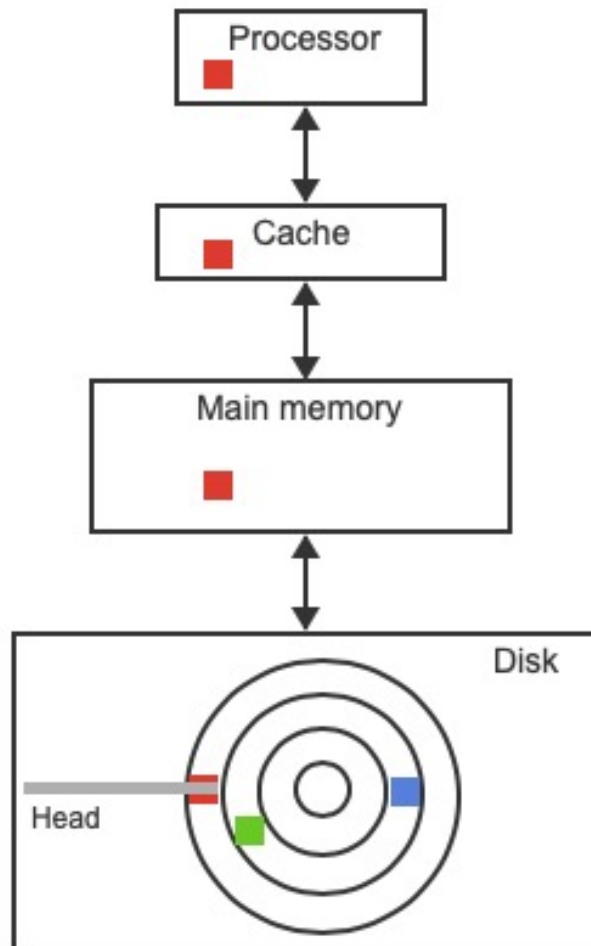
P&H Ch 5

**COMP 122: Computer
Architecture and
Assembly Language**
Spring 2020

5.2.1: Primary technologies used in memories.

] 2x speed

Memory
hierarchy



Current technology

SRAM

DRAM

Magnetic disk

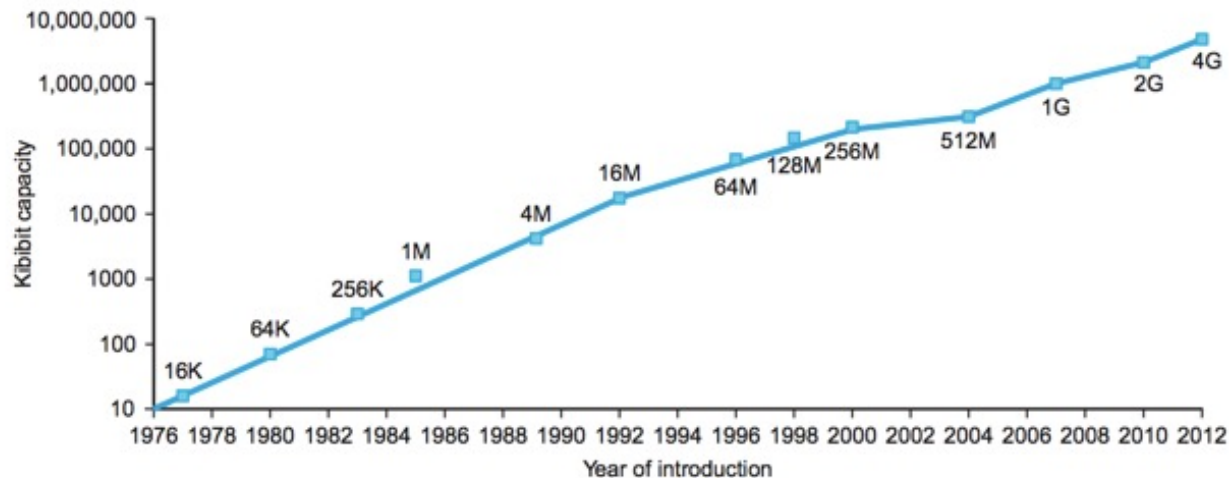
** Flash memory used in
personal mobile devices*

DRAM Timeline

Hennessy & Patterson

Figure 1.5.1: Growth of capacity per DRAM chip over time (COD Figure 1.11).

The y-axis is measured in kibibits (2^{10} bits). The DRAM industry quadrupled capacity almost every three years, a 60% increase per year, for 20 years. In recent years, the rate has slowed down and is somewhat closer to doubling every two years to three years.



Refresh rate [[edit](#)]

Main article: [Memory refresh](#)

See also: [§ Security](#)

64ms

Typically, manufacturers specify that each row must be refreshed every 64 ms or less, as defined by the [JEDEC](#) standard.

Some systems refresh every row in a burst of activity involving all rows every 64 ms. Other systems refresh one row at a time staggered throughout the 64 ms interval. For example, a system with $2^{13} = 8,192$ rows would require a staggered [refresh rate](#) of one row every $7.8 \mu\text{s}$ which is 64 ms divided by 8,192 rows. A few real-time systems refresh

DRAM History

the cost advantage increased; the 16 kbit Mostek MK4116 DRAM,^{[17][18]} introduced in 1976, achieved greater than 75% worldwide DRAM market share. However, as density increased to 64 kbit in the early 1980s, Mostek and other US manufacturers were overtaken by Japanese DRAM manufacturers, which dominated the US and worldwide markets during the 1980s and 1990s.

Early in 1985, [Gordon Moore](#) decided to withdraw Intel from producing DRAM.^[19] By 1986, all United States chip makers had stopped making DRAMs.^[20]

In 1985, when 64K DRAM memory chips were the most common memory chips used in computers, and when more than 60 percent of those chips were produced by Japanese companies, semiconductor makers in the United States accused Japanese companies of [export dumping](#) for the purpose of driving makers in the United States out of the commodity memory chip business.^[21]

[Synchronous dynamic random-access memory](#) (SDRAM) was developed by [Samsung](#). The first commercial SDRAM chip was the Samsung KM48SL2000, which had a capacity of 16 [Mb](#),^[22] and was introduced in 1992.^[23] The first commercial [DDR SDRAM](#) ([double data rate SDRAM](#)) memory chip was Samsung's 64 Mb DDR SDRAM chip, released in 1998.^[24]

Later, in 2001, Japanese DRAM makers accused Korean DRAM manufacturers of dumping.^[25]

In 2002, US computer makers made claims of [DRAM price fixing](#).

Memory

DRAM

P&H Ch 5

**COMP 122: Computer
Architecture and
Assembly Language**
Spring 2020

Figure 5.2.1: DRAM size increased by multiples of four approximately once every three years until 1996 and thereafter considerably slower (COD Figure 5.5).

Year introduced	Chip size	\$ per GiB	Total access time to a new row/column	Average column access time to existing row
1980	64 Kibibit	\$1,500,000	250 ns	150 ns
1983	256 Kibibit	\$500,000	185 ns	100 ns
1985	1 Mebibit	\$200,000	135 ns	40 ns
1989	4 Mebibit	\$50,000	110 ns	40 ns
1992	16 Mebibit	\$15,000	90 ns	30 ns
1996	64 Mebibit	\$10,000	60 ns	12 ns
1998	128 Mebibit	\$4,000	60 ns	10 ns
2000	256 Mebibit	\$1,000	55 ns	7 ns
2004	512 Mebibit	\$250	50 ns	5 ns
2007	1 Gibibit	\$50	45 ns	1.25 ns
2010	2 Gibibit	\$30	40 ns	1 ns
2012	4 Gibibit	\$1	35 ns	0.8 ns

Micron DRAM



Our History of Innovation

1978

1979

1981

1984

1987



Micron Technology Is Founded

Micron started as a four-person semiconductor design company in the basement of a Boise, Idaho dental office. Micron's first contract was for the design of a 64K memory chip for Mostek Corporation.

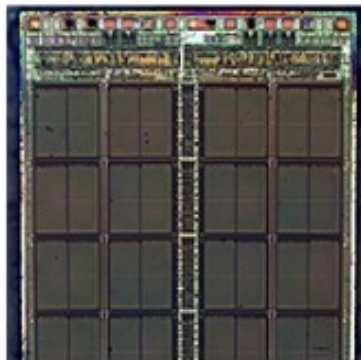
64Kb

1Mb

1987

Micron Brings 1-Megabit DRAM to Market

A milestone in density, the 1Mb DRAM became a staple for main memory in PCs and graphics cards during the late 1980s and 1990s. Micron's 1Mb DRAM enabled high-capacity SIMM modules that supported PCs equipped with Microsoft's new Windows OS.



Micron DDR DRAM

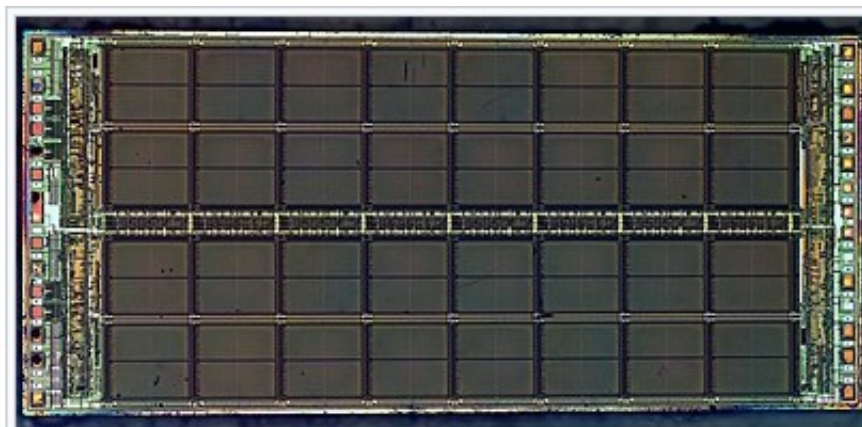


Double Data Rate

Founded: May 22, 1978

Revenue: \$21.44 billion (2020)

Headquarters: Boise, ID



A die photograph of the [Micron Technology MT4C1024 DRAM integrated circuit](#). It has a capacity of 1 [megabit](#) equivalent of 2^{20} bits or 128 kB.^[1]

Micron DRAM

By Technology

View part catalogs, download data sheets, and find other product information.

DDR5 SDRAM



DDR SDRAM



DDR4 SDRAM



SDRAM



DDR3 SDRAM



RLDRAM Memory



DDR2 SDRAM



LPDRAM



RAM/ROM (x8) Chips

MCS-8

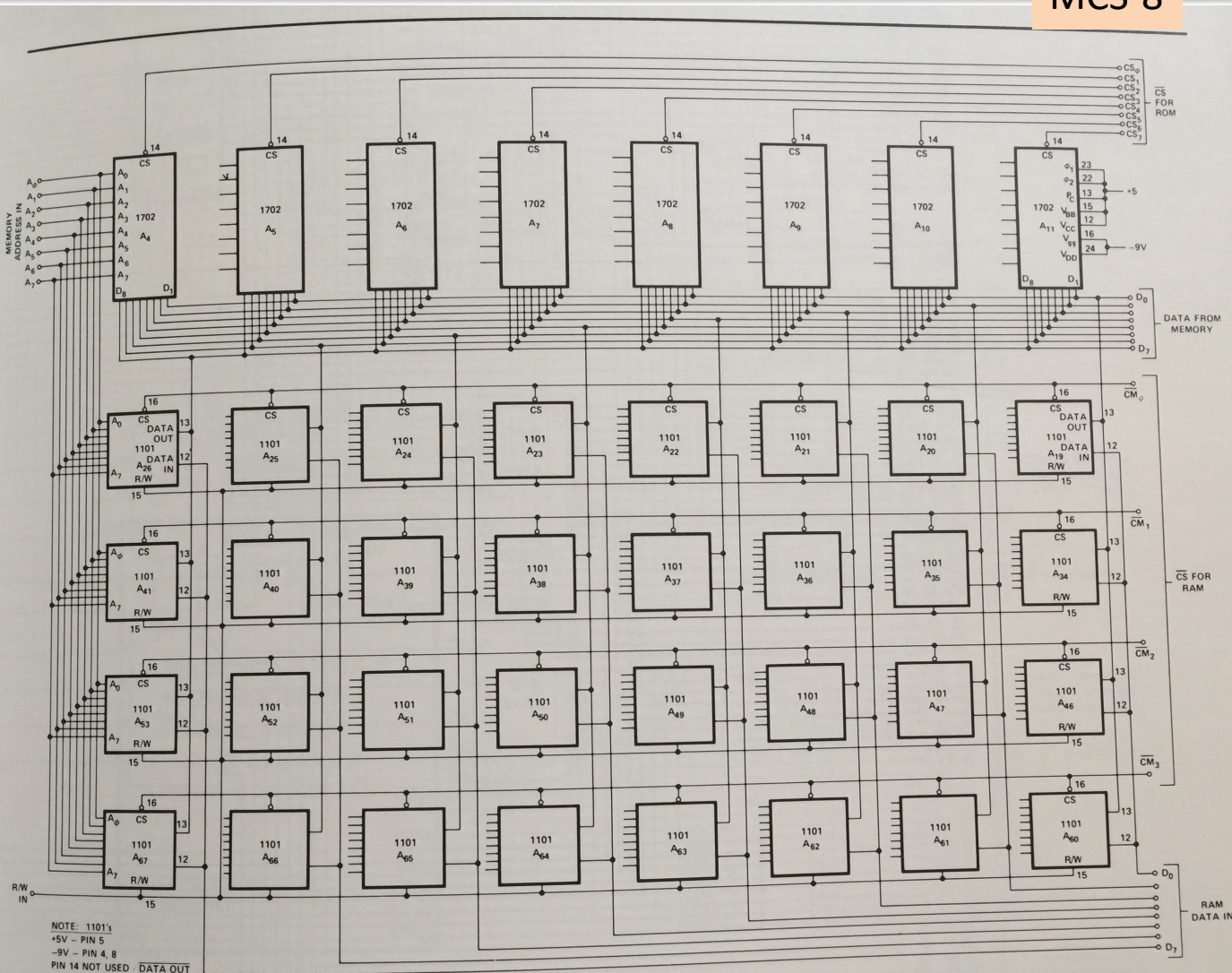


Figure 9. MCS-8 Memory System

WOM!

April 1, 1980

Signetics

FULLY ENCODED, 9046xN, RANDOM ACCESS
WRITE-ONLY-MEMORY

25120

Do Not Copy

FINAL SPECIFICATION(10)

DESCRIPTION

The Signetics 25000 Series 9046XN Random Access Write-Only-Memory employs both enhancement and depletion mode P-Channel, N-Channel, and neu⁽¹⁾ channel MOS devices. Although a static device, a single TTL level clock phase is required to drive the on-board multi-port clock generator. Data refresh is accomplished during CB and LH periods⁽¹¹⁾. Quadri-state outputs (when applicable) allow expansion in many directions, depending on organization.

The static memory cells are operated dynamically to yield extremely low power dissipation. All inputs and outputs are directly TTL compatible when proper interfacing circuitry is employed.

Device construction is more or less S.O.S.⁽²⁾.

FEATURES

- FULLY ENCODED MULTI-PORT ADDRESSING
- WRITE CYCLE TIME 80nS (MAX. TYPICAL)
- WRITE ACCESS TIME⁽³⁾
- POWER DISSIPATION 10uW/BIT TYPICAL
- CELL REFRESH TIME 2mS (MIN. TYPICAL)

BIPOLAR COMPATIBILITY

All data and clock inputs plus applicable outputs will interface directly or nearly directly with bipolar circuits of suitable characteristics. In any event use 1 amp fuses in all power supply and data lines.

INPUT PROTECTION

All terminals are provided with slip-on latex protectors for the prevention of Voltage Destruction. (PILL packaged devices do not require protection).

SILICON PACKAGING

Low cost silicon DIP packaging is implemented and reliability is assured by the use of a non-hermetic sealing technique which prevents the entrapment of harmful ions, but which allows the free exchange of friendly ions.

SPECIAL FEATURES

Because of the employment of the Signetics' proprietary Sanderson-Rabbet Channel the 25120 will provide 50% higher speed than you will obtain.

COOLING

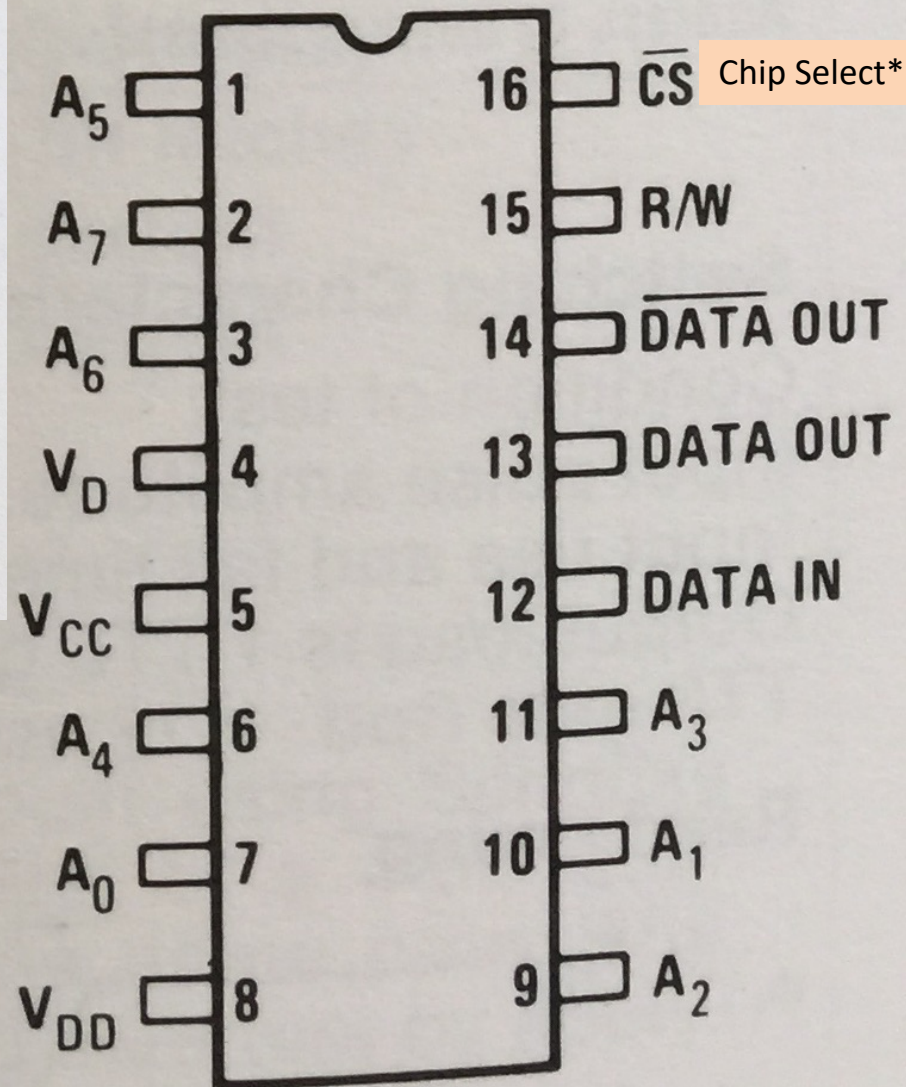
The 25120 is easily cooled by employment of a six-foot

i1101A 256x1 SRAM

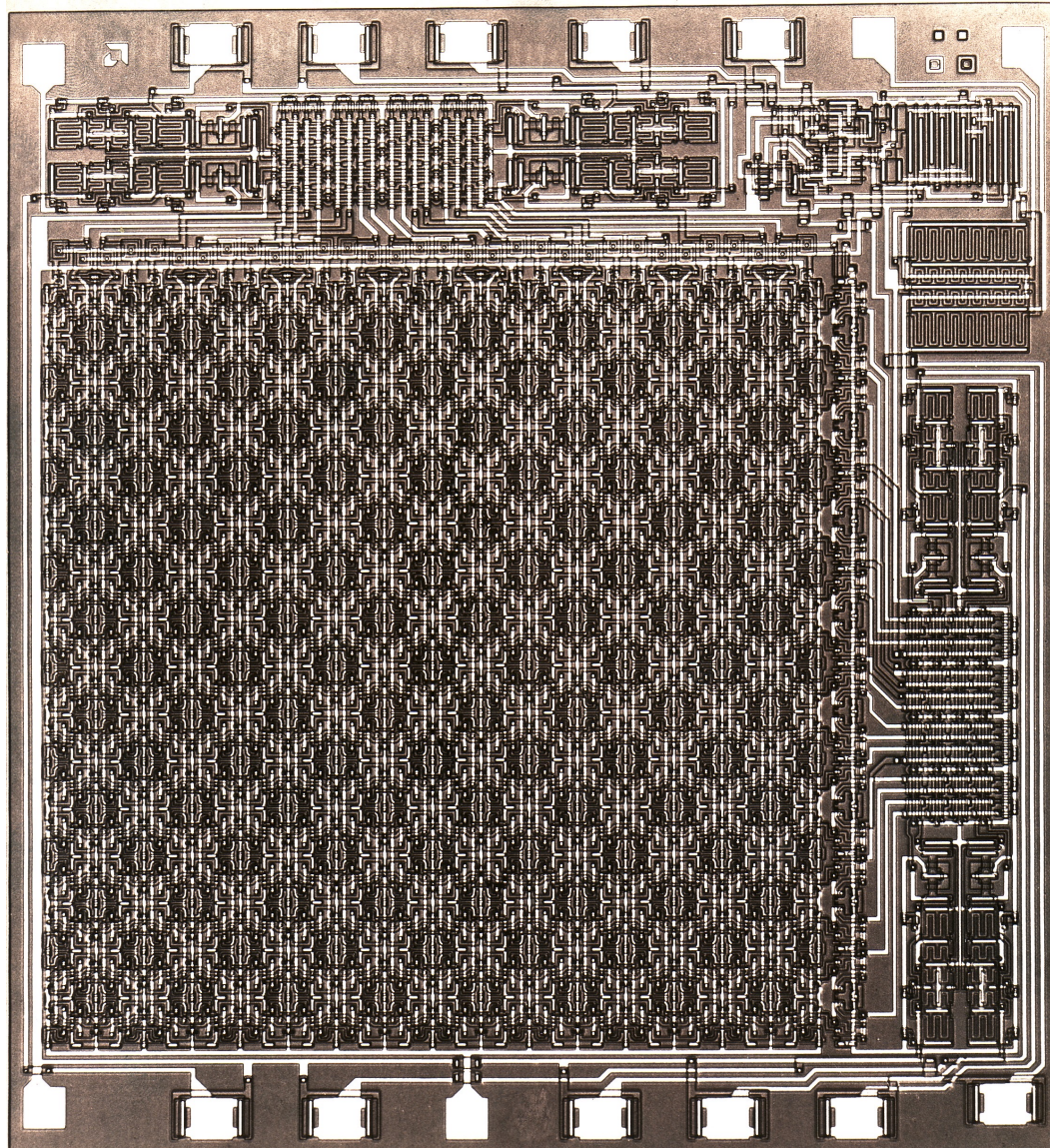
Pinout

MCS-8

- Access time below 750 ns typically, 1.0 μ sec maximum — 1101A1; 1.5 μ sec maximum — 1101A: over temperature
- Low power dissipation—typically less than 1.5 mW/bit during access
- Low power standby mode
- Directly DTL and TTL compatible
- OR-Tie capability
- Simple memory expansion—chip-select input lead
- Fully decoded—on-chip address decode and sense
- Inputs protected against static charge
- Ceramic and plastic package
- Silicon gate MOS technology



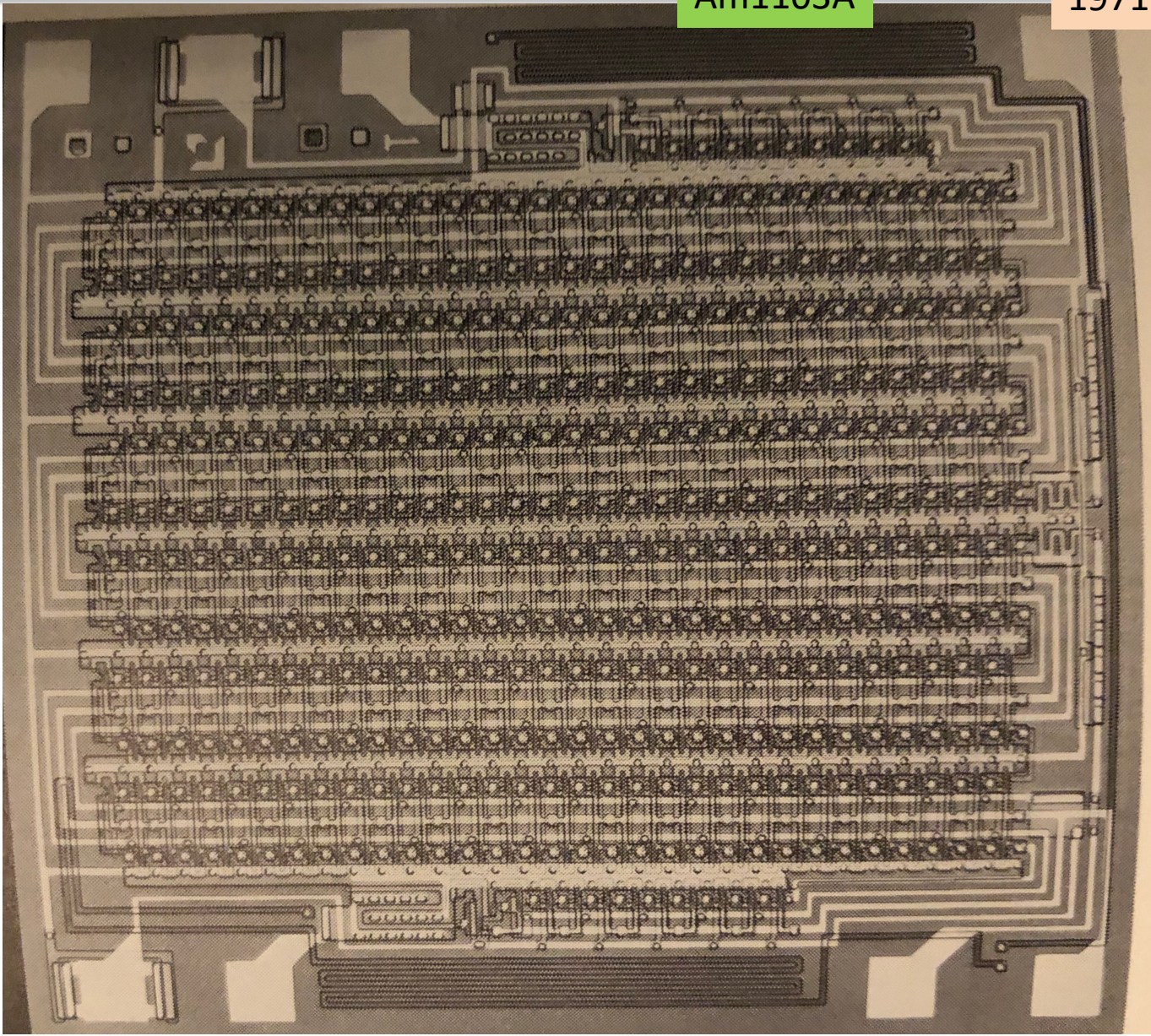
Am1101A 256x1 SRAM



Am1103 1Kx1 DRAM

Am1103A

1971



AMD 1K DRAM's

DRAM

256x4 Organization

4-bit
Data
Bus

Part Number	No. of Pins	Worst Case Access Time	I/O	Chip Enable
Am9101A	22	500 nsec	Separate	2
Am9101B	22	400 nsec	Separate	2
Am9101C	22	300 nsec	Separate	2
Am9101D	22	250 nsec	Separate	2
Am9111A	18	500 nsec	Common	2
Am9111B	18	400 nsec	Common	2
Am9111C	18	300 nsec	Common	2
Am9111D	18	250 nsec	Common	2

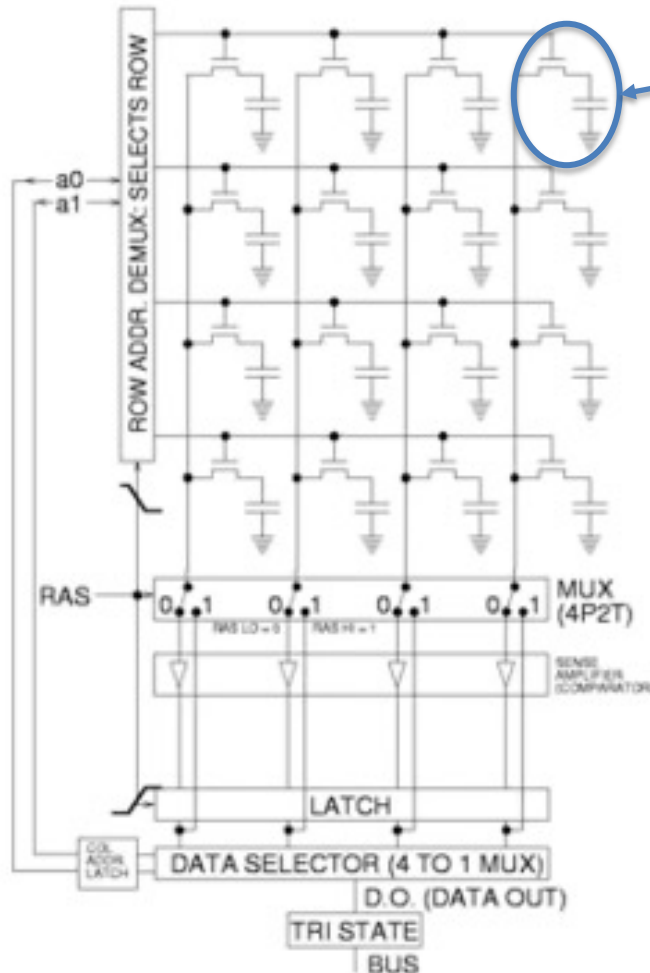
Chip Select*

1K x1

Number	Package	Time	Range
Am9102DM	Hermetic DIP	650 nsec	303 mw
Am9102ADM	Hermetic DIP	500 nsec	303 mw
Am9102BDM	Hermetic DIP	400 nsec	303 mw
Am9102CDM	Hermetic DIP	300 nsec	330 mw
Am91L02DM	Hermetic DIP	650 nsec	193 mw
Am91L02ADM	Hermetic DIP	500 nsec	193 mw
Am91L02BDM	Hermetic DIP	400 nsec	193 mw
Am91L02CDM	Hermetic DIP	300 nsec	204 mw
Am9102FM	Flat Package	650 nsec	303 mw
Am9102AFM	Flat Package	500 nsec	303 mw

DRAM Schematic

The DRAM

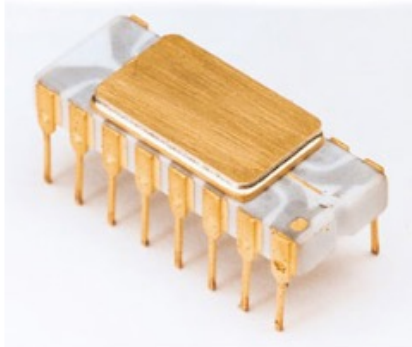


- ❖ 1 transistor
- ❖ 1 cap (parasitic)

MPU/MCU Museum

[Nostalgia Dept.]

Intel 4004 is announced, November 15, 1971

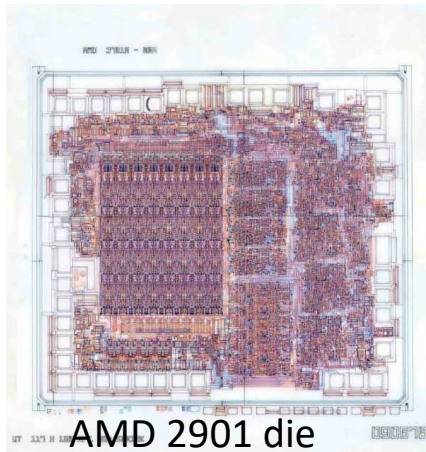


The building-block 4004 CPU held 2300 transistors. The microprocessor, the size of a little fingernail, delivered the same computing power as the first electronic computer built in 1946, which, in contrast, filled a room.

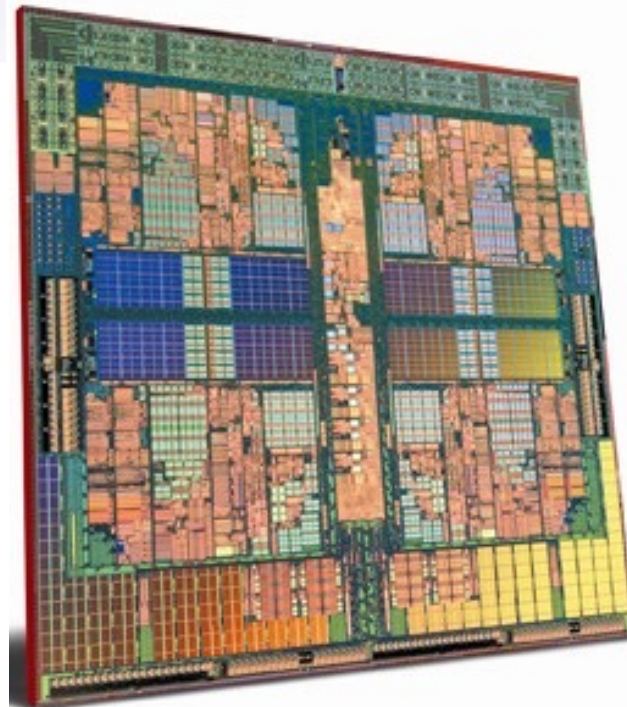
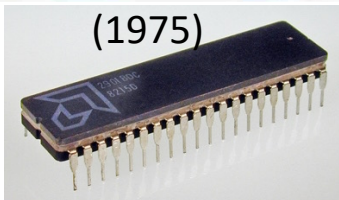
From: EDN Nov 20, 2015

[Intel 4004 is announced, November 15, 1971](#)

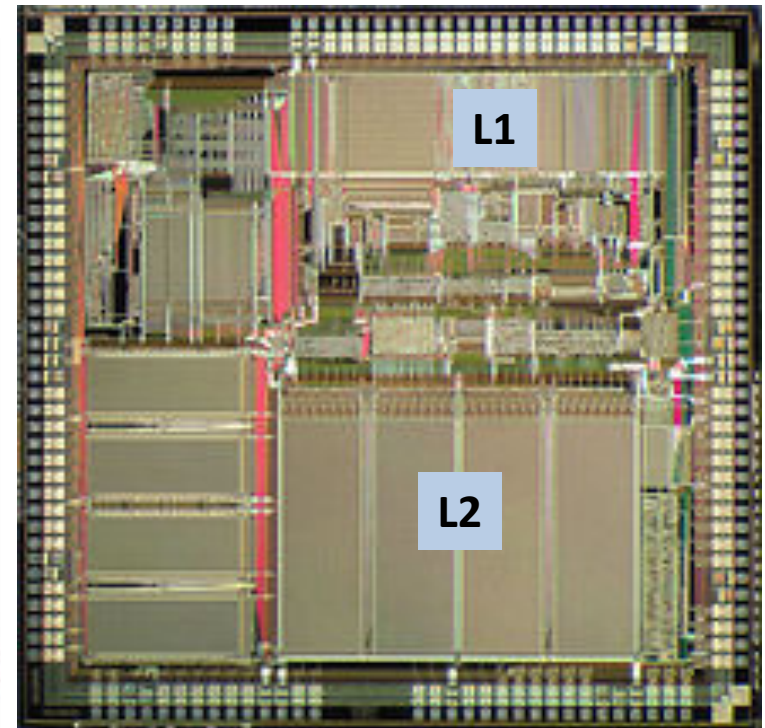
The venerable 16-pin side-braced DIP.
(Click image to view full size)



AMD 2901 die
(1975)



AMD quad-core die



ARM610 sgl-core die

Memory Museum

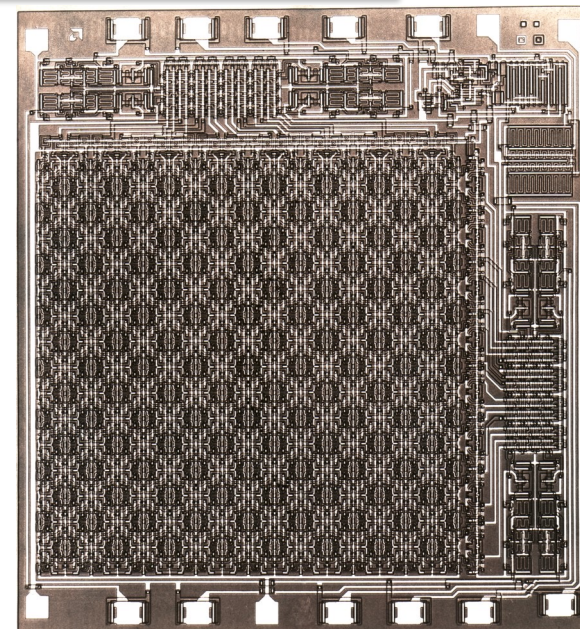
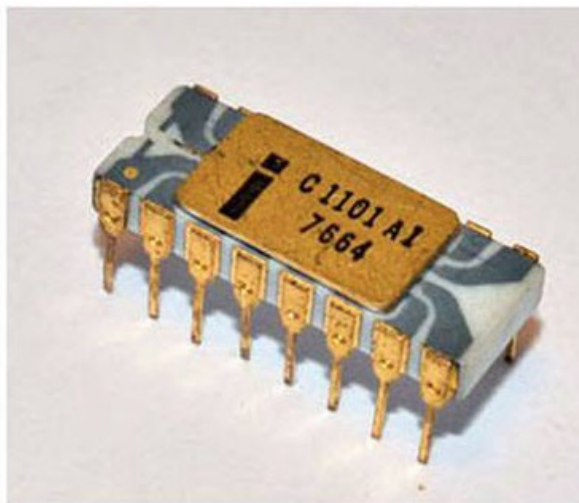
COMP122

July 1969.

Intel 1101: The first MOS memory chip.

A 256-bit SRAM (Static Random Access Memory).

The 1101 was developed in parallel with the 3101, but lost the race to be Intel's first product. It was produced with Silicon gate MOS (Metal Oxide Semiconductor) technology, which gave Intel the edge it needed to produce high density memories.



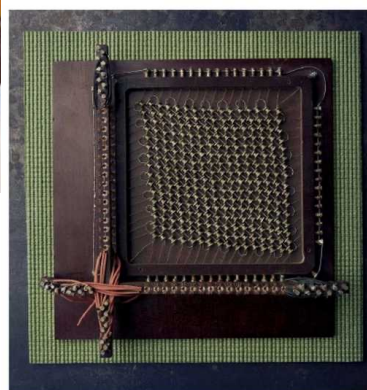
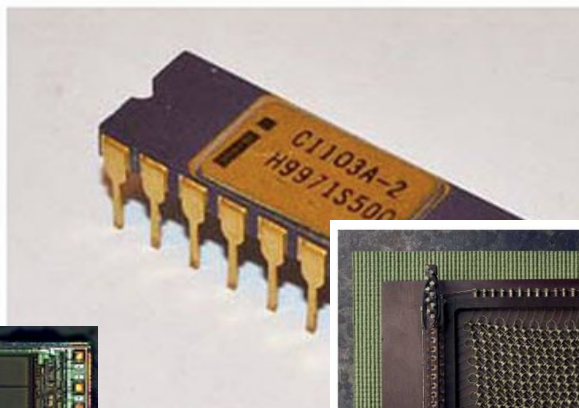
Am1101A

October 1970.

Intel 1103: The first DRAM memory chip.

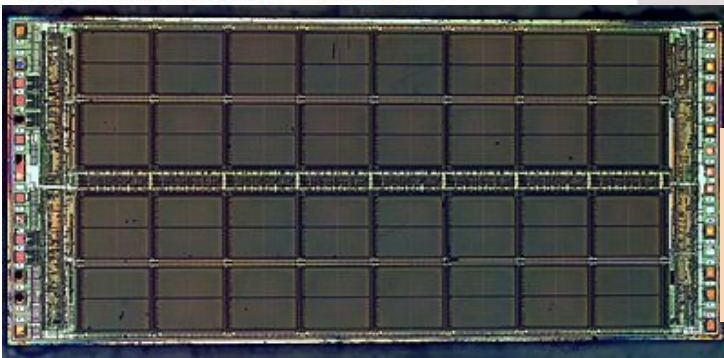
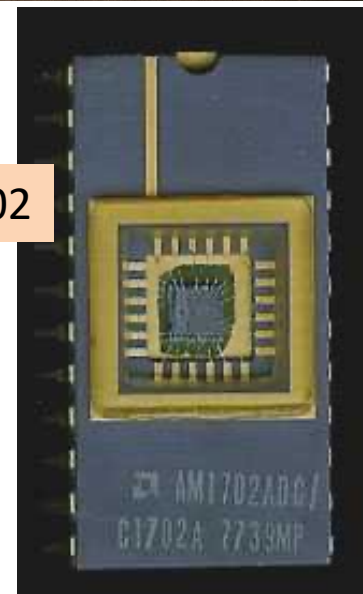
A 1024-bit DRAM (Dynamic Random Access Memory).

This is the chip that kicked magnetic [Core Memory](#) out of the game, making Intel a world leader in memories for a decade.



256-bit magnetic core memory (c. 1952) Slow data retrieval and storage speeds limited the utility of early computers. RCA researcher Jan Rajchman's solution was a memory array consisting of a wire matrix with doughnut-shaped magnetic cores at each intersection. By applying a current to a given set of horizontal and vertical wires, you could select a specific core and quickly change the direction of its magnetic field.

Am1702



MT4C
1Mb
DRAM

Memory Chips

DRAM 1T



Dynamic random-access memory (DRAM) is a type of random access semiconductor memory that stores each bit of data in a memory cell consisting of a tiny capacitor and a transistor, typically a MOSFET. The capacitor can either be charged or discharged; these two states are taken to

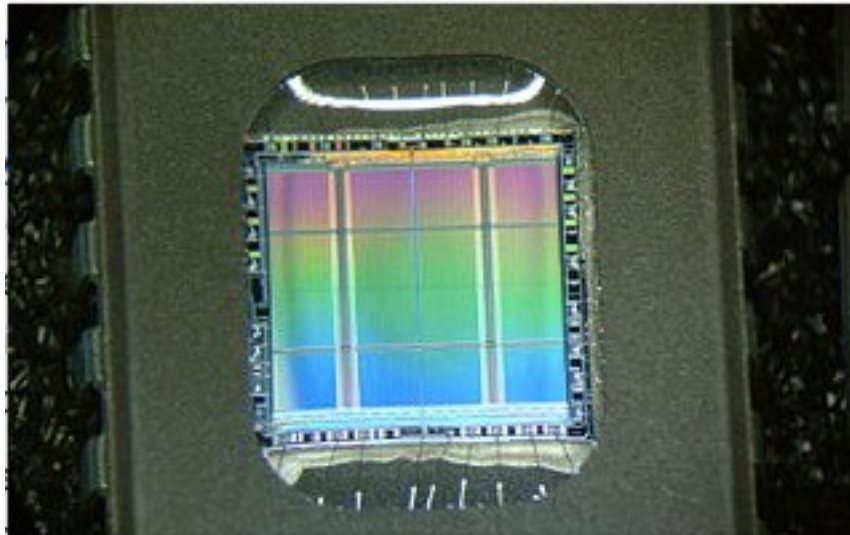
SRAM 4T



Static random-access memory is a type of semiconductor random-access memory (RAM) that uses bistable latching circuitry (flip-flop) to store each bit. SRAM exhibits data remanence, but it is still *volatile* in the conventional sense that data is eventually lost when the memory is not powered.

Memory Chips

ROM

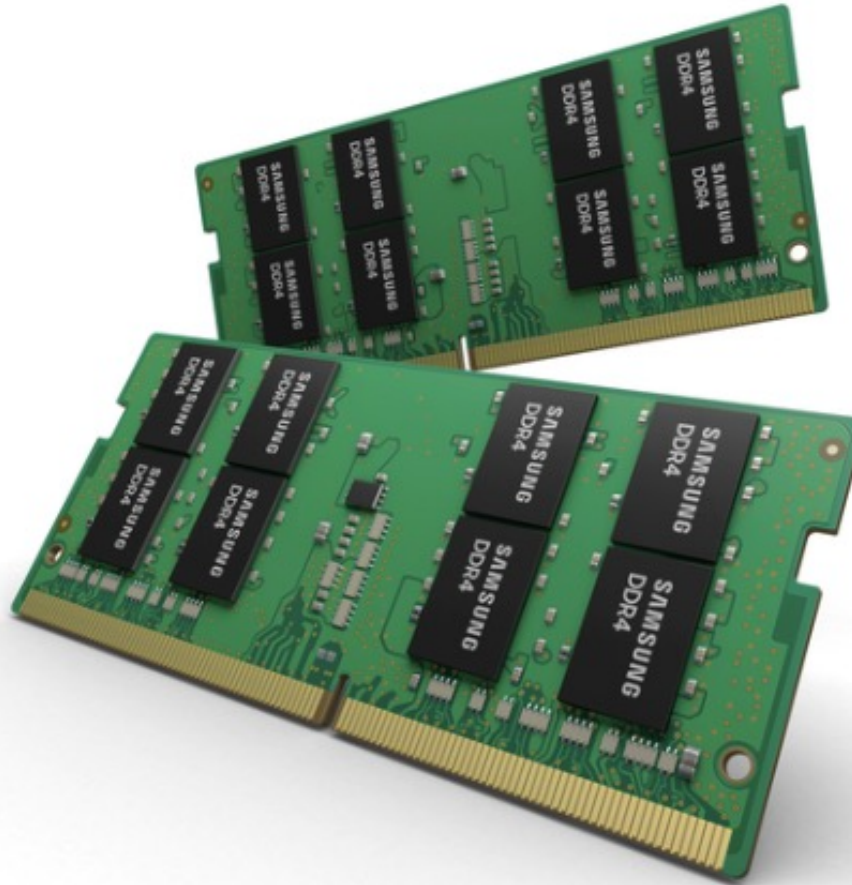


Read-only memory (ROM) is a type of non-volatile memory used in computers and other electronic devices. Data stored in ROM cannot be electronically modified after the manufacture of the memory device. Read-only memory is useful for storing software that is rarely changed during the life of the system.

- ❖ ROM (masked)
- ❖ PROM
- ❖ EPROM
- ❖ EEPROM
- ❖ Flash E²

Modern Memory

32GiB DRAM



Anyway, back to earth! Here are a pair of 32GiB DDR4 SO-DIMMs. These run around \$200 each, so for 1TB, you're talking \$6,400 using this kind of memory. You're also going to need a pretty high end CPU to address that much memory (and it's unlikely it would be using SO-DIMMs, they're a laptop thing in general),

Modern Memory

64GiB DRAM



A standard unbuffered DIMM for desktop PCs in 64GiB capacity will run around \$400, so not a big win here, either.

DDR SDRAM

Samsung

COMP122

Double data rate synchronous DRAM [edit]

Main articles: [DDR SDRAM](#), [DDR2 SDRAM](#), [DDR3 SDRAM](#), and [DDR4 SDRAM](#)

Double data rate SDRAM (DDR SDRAM or DDR) was a later development of SDRAM, used in PC memory beginning in 2000. Subsequent versions are numbered sequentially (*DDR2*, *DDR3*, etc.). DDR SDRAM internally performs double-width accesses at the clock rate, and uses a [double data rate](#) interface to transfer one half on each clock edge. DDR2 and DDR3 increased this factor to 4x and 8x, respectively, delivering 4-word and 8-word bursts over 2 and 4 clock cycles, respectively. The internal access rate is mostly unchanged (200 million per second for DDR-400, DDR2-800 and DDR3-1600 memory), but each access transfers more data.

Direct Rambus DRAM [edit]

Main article: [RDRAM](#)

Direct Rambus DRAM (RDRAM) was developed by Rambus. First supported on motherboards in 1999, it was intended

Synchronous dynamic RAM [edit]

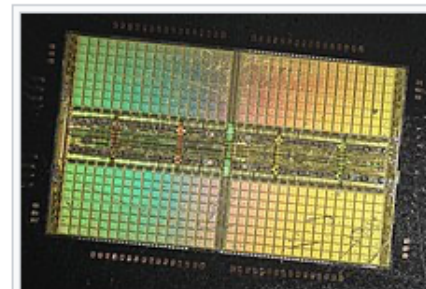
Main article: [Synchronous dynamic random-access memory](#)

Synchronous dynamic RAM (SDRAM) significantly revises the asynchronous memory interface, adding a clock (and a clock enable) line. All other signals are received on the rising edge of the clock.

The $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ inputs no longer act as strobes, but are instead, along with $\overline{\text{WE}}$, part of a 3-bit command:

SDRAM Command summary

CS	RAS	CAS	WE	Address	Command
H	x	x	x	x	Command inhibit (no operation)
L	H	H	H	x	No operation
L	H	H	L	x	Burst Terminate: stop a read or write burst in progress.
L	H	L	H	Column	Read from currently active row.
L	H	L	L	Column	Write to currently active row.
L	L	H	H	Row	Activate a row for read and write.
L	L	H	L	x	Precharge (deactivate) the current row.
L	L	L	H	x	Auto refresh: refresh one row of each bank, using an internal counter.
L	L	L	L	Mode	Load mode register: address bus specifies DRAM operation mode.



The die of a Samsung DDR-SDRAM 64MBit package

DDR SDRAM

Synchronous DRAM Double Data Rate

What is GDDR6 SDRAM?



Jeff Drobman · just now

Lecturer at California State University, Northridge (2016–present)

6th generation sync DRAM for graphics. now a JEDEC standard:

"This document defines the Graphics Double Data Rate 6 (GDDR6) Synchronous Graphics Random Access Memory (SGRAM) specification, including features, functionality, package, and pin assignments. The purpose of this Specification is to define the minimum set of requirements for 8 Gb through 16 Gb x16 dual channel GDDR6 SGRAM devices."

Synchronous dynamic random-access memory (SDRAM) is any dynamic random-access memory (DRAM) where the operation of its external pin interface is coordinated by an externally supplied clock signal.

Synchronous = Clocked

Register (pipelined)



GDDR6 SDRAM

From Wikipedia, the free encyclopedia

This article is about DDR graphics RAM (SGRAM). For non-graphics (dynamic) DDR memory, see [SDRAM](#).

GDDR6 SDRAM (**Graphics Double Data Rate 6 Synchronous Dynamic Random-Access Memory**) is a type of [synchronous graphics random-access memory](#) (SGRAM) with a high [bandwidth](#) ("double data rate") interface designed for use in [graphics cards](#), [game consoles](#), and [high-performance computing](#). It is a type of [GDDR SDRAM](#) (graphics [DDR SDRAM](#)), and is the successor to [GDDR5](#). Just like GDDR5X and despite its name it uses QDR (quad data rate).^[1]

Overview [\[edit \]](#)

The finalised specification was published by [JEDEC](#) in July 2017.^[2] GDDR6 offers increased per-pin bandwidth (up to 16 [Gbit/s](#)^[3]) and lower operating voltages (1.35 V^[4]), increasing performance and decreasing power consumption relative to [GDDR5X](#).^{[5][6]}

Commercial implementation [\[edit \]](#)

At [Hot Chips 2016](#), [Samsung](#) announced GDDR6 as the successor of [GDDR5X](#).^{[5][6]} Samsung later announced that the first products would be 16 Gbit/s, 1.35 V chips.^{[7][8]} In January 2018, Samsung began mass production of 16 [Gb](#) (2 [GB](#)) GDDR6 chips, fabricated on a 10 nm class [FinFET](#) process and with a data rate of up to 18 Gbit/s per pin.^{[9][8][10]}

In February 2017, [Micron Technology](#) announced it would release its own GDDR6 products by early 2018.^[11] Micron began mass production of 8 Gb chips in June 2018.^[12]

Micron

[SK Hynix](#) announced its GDDR6 products would be released in early 2018.^{[13][3]} SK Hynix announced in April 2017 that its GDDR6 chips would be produced on a 21 nm process and be 10% lower voltage than GDDR5.^[3] The SK Hynix chips were expected to have a transfer rate of 14–16 Gbit/s.^[4] The first graphics cards to use SK Hynix's GDDR6 RAM were expected to use 12 GB of RAM with a 384-bit memory bus, yielding a bandwidth of 768 GB/s.^[3] SK Hynix began mass production in February 2018, with 8 Gbit chips and a data rate of 14 Gbit/s per pin.^[14]

DDR SDRAM

Graphics

Synchronous DRAM

Double Data Rate

Nvidia + AMD

[Nvidia](#) officially announced the first consumer graphics cards using GDDR6, the [Turing](#)-based [GeForce RTX 2080 Ti](#), RTX 2080 & RTX 2070 on August 20, 2018,^[15] RTX 2060 on January 6, 2019^[16] and GTX 1660 Ti on February 22, 2019.^[17] GDDR6 memory from [Samsung Electronics](#) is also used for the Turing-based [Quadro RTX](#) series.^[18] The RTX 20 series initially launched with Micron memory chips, before switching to Samsung chips by November 2018.^[19]

[AMD](#) officially announced the [Radeon RX 5700](#), 5700 XT, and 5700 XT 50th Anniversary Edition on June 10, 2019. These Navi 10^[20] GPUs utilize 8 GB of GDDR6 memory.^[21]

Micron

GDDR6X [\[edit \]](#)

Micron developed GDDR6X in Close Collaboration with Nvidia. GDDR6X SGRAM had not been standardized by JEDEC yet. Nvidia is Micron's only GDDR6X launch partner.^[22] GDDR6X offers increased per-pin bandwidth between 19-21 Gbit/s with [PAM4](#) signaling, allowing two bits to be transmitted per transmission, replacing earlier [NRZ](#) (non return to zero, PAM2) coding which only allowed for a bit per transmission, which limited the per-pin bandwidth of GDDR6 to 16 Gbit/s.^[23] The first to use GDDR6X are the [Nvidia RTX 3080 and 3090](#) graphics cards. PAM4 signalling is not new but it costs more to implement, partly because it requires more space and is more prone to signal-to-noise ratio (SNR) issues,^[24] which mostly limited its use to high speed networking (like 200G Ethernet). GDDR6X consumes 15% less power per transferred bit than GDDR6, but power consumption is higher since GDDR6X is faster than GDDR6. PAM4 consumes less power and pins than differential signalling while still being faster than NRZ. GDDR6X is thought to be cheaper than [High Bandwidth Memory](#).^[25]

DDR4 Vulnerability

FURTHER READING

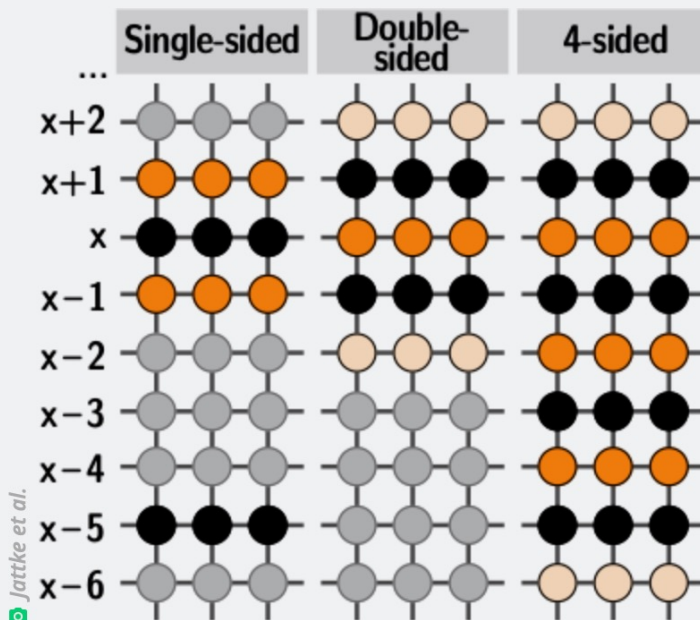
Once thought safe, DDR4 shown to be vulnerable to "Rowhammer"



All previous Rowhammer attacks have hammered rows with uniform patterns, such as single-sided, double-sided, or n-sided. In all three cases, these "aggressor" rows—meaning those that cause bitflips in nearby "victim" rows—are accessed the same number of times.

- ❖ Attacks
- ❖ Exploits

➤ Rowhammer

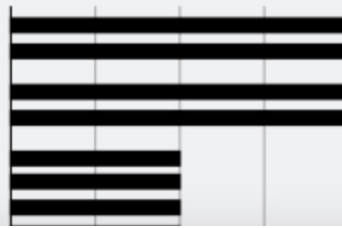


Rowhammer access patterns from previous work, showing spatial arrangement of aggressor rows (in black) and victim rows (in orange and cream) in DRAM memory.

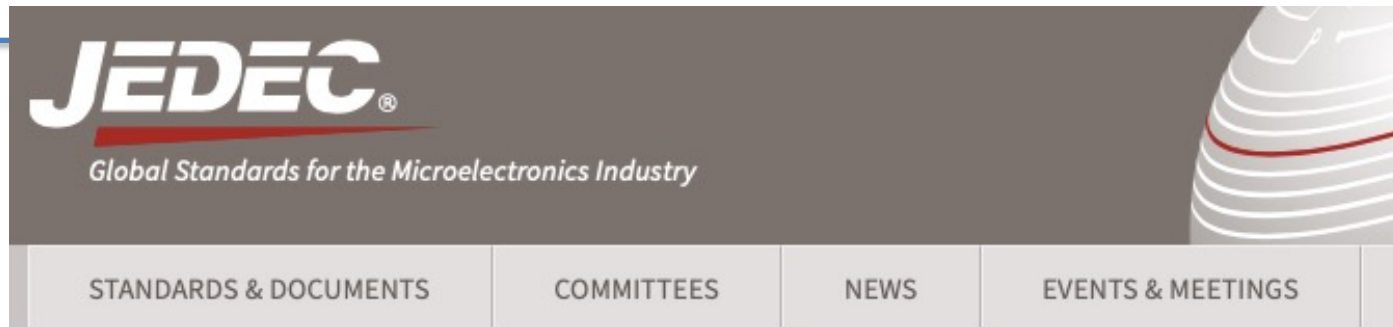
Single-sided

Double-sided

4-sided



JEDEC (EIA)



GRAPHICS DOUBLE DATA RATE 6 (GDDR6) SGRAM STANDARD

JESD250B

Published: Nov 2018

This document defines the Graphics Double Data Rate 6 (GDDR6) Synchronous Graphics Random Access Memory (SGRAM) specification, including features, functionality, package, and pin assignments. The purpose of this Specification is to define the minimum set of requirements for 8 Gb through 16 Gb x16 dual channel GDDR6 SGRAM devices. System designs based on the required aspects of this standard will be supported by all GDDR6 SGRAM vendors providing compatible devices. Some aspects of the GDDR6 standard such as AC timings and capacitance values were not standardized. Some features are optional and therefore may vary among vendors. In all cases, vendor data sheets should be consulted for specifics. This document was created based on some aspects of the GDDR5 Standard (JESD212). Item 1836.99D.

JEDEC (EIA)

JEDEC History

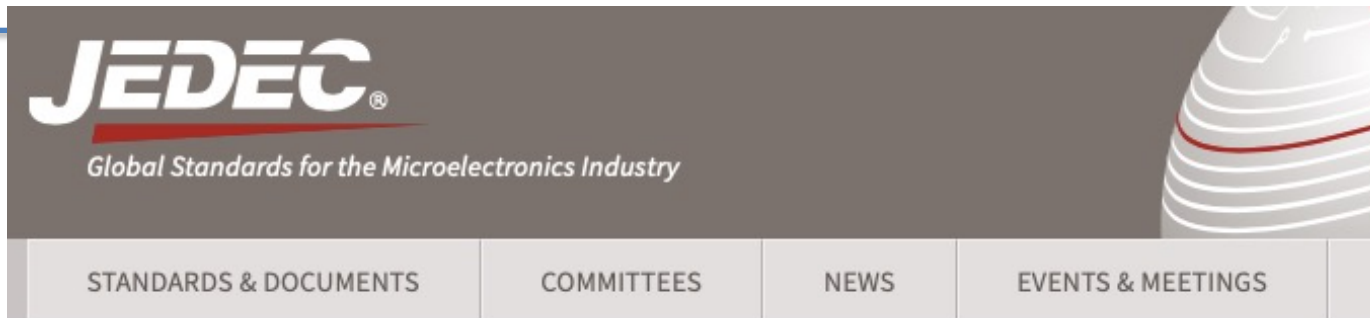
In 1924, the Radio Manufacturers Association (which later became the Electronic Industries Association) was established. In 1944, the Radio Manufacturers Association and the National Electronic Manufacturers Association established the Joint Electron Tube Engineering Council (JETEC), which was responsible for assigning and coordinating type numbers of electron tubes. As the radio industry expanded into the emerging field of electronics, various divisions of the EIA, including JETEC, began to function as semi-independent membership groups. The Council expanded its scope to include solid state devices, and by 1958 the organization was renamed the Joint Electron Device Engineering Council (JEDEC) – one council for tubes and one for semiconductors.

Timeline

- [Pre-1960s](#)
- [1960s](#)
- [1970s](#)
- [1980s](#)
- [1990s](#)
- [2000s](#)
- [2010s](#)

JEDEC initially functioned within the engineering department of EIA where its primary activity was to develop and assign part numbers to devices. Over the next 50 years, JEDEC's work expanded into developing test methods and product standards that proved vital to the development of the semiconductor industry. Among the landmark standards that have come from JEDEC committees are:

JEDEC (EIA)



Why JEDEC Standards Matter

JEDEC committees develop open standards, which are the basic building blocks of the digital economy and form the bedrock on which healthy, high-volume markets are built. For example, JEDEC semiconductor memory standards - from dynamic RAM chips and memory modules to DDR synchronous DRAM and flash components - have enabled huge markets in PCs, servers, digital cameras, MP3 players, smart phones, automotive and HDTV, to name just a few.

Standards enable innovation, serving to commoditize components by lowering their prices while maintaining quality and reliability. This leads suppliers to compete more vigorously on innovative features and gives buyers more variety and a broader selection. The end result is a much larger market than proprietary products could foster, which means more potential sales and revenue.

Standards allow companies to invest more strategically in R&D rather than inventing everything from scratch. Once common form factors are set, companies can base their designs on standards and focus on innovation.

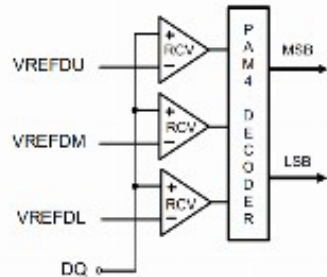
Micron (DRAM) News

COMP122

PAM-4: 4-level data encoding on analog signals

Technology in Science and Industry

Micron Brings 4-level Logic To Market



--Micron

4-level sense circuit for Micron GDDR6X

Micron Technology has brought PAM4 signaling to high-speed DRAM in a release that may well be the dawn of the multi-level logic era.

Finally, however, a new option is becoming practical – increasing the number of bits per pin.

Traditional memory interfaces have used the same kinds of signaling that standard logic uses, a non-return to zero (NRZ) binary with two signal levels conveying one bit. Networking and other high-speed buses like 56G Ethernet, however, began switching to a multi-level signal some years ago. Their four-level pulse amplitude modulation (PAM4) signaling scheme encodes two bits per signal line and a number of transceivers are currently available that support this scheme.

More: [PAM4 Makes it to Memory Interfaces](#)

From EDN -- Contributed by John Springer on 22 September 2020

Jeff Drobman - From Chatsworth, CA. Posted 26 Sep 2020

it seems it is merely a 4-level analog signal encoding of data, which gets decoded into a conventional 2 bits. this is a means of doubling bit rates.

100BASE-TX Ethernet has long used "MLT-3" encoding as a 3-level signal (+5v, 0v, -5v) form of NRZ. NRZ is a means of reducing the effective baseband signal frequency. problem with that encoding is it is baseband and suffers from baseline wander (up to .75V). (I was applications mgr for my company on that product. Lucent referenced my app note on their patent.)

Modern Memory

64GB Flash



➤ Used on Raspberry Pi 4



A disassembled **USB flash drive**. The chip on the left is flash memory. The **controller** is on the right.

Here's a 64GB microSD card. I found this on Amazon for \$12.49 just now... directly from Amazon and Sandisk, too, so it's not even a fake. Why so cheap? Well, flash memory is much slower and inherently much cheaper than DRAM. It's higher density, and more space efficient. This is probably using 3D memory chips, with each floating gate transistor storing 3-bits of data (TLC Flash) and storage stacks 40+ devices high. An SD-card has a 4-bit-wide bus, a DDR-4 DIMM has a 64-bit-wide bus.

Modern Memory

1TB Flash



Or just buy one of these M.2 SSDs for just over \$100. Flash is just inherently cheaper than DRAM. Or for the same price, maybe an 8TB hard drive. Magnetic storage is cheaper still.

And practically no individual needs 64GiB of DRAM, much less 1TiB. Sure, very large servers, very large databases may, but that isn't usually a thing you have at home, or in a personal computer. What are you going to do with all that DRAM?

Memory Errors

P&H Ch 5

COMP 122: Computer
Architecture and
Assembly Language
Spring 2020

What is a parity checker?



Jeff Drobman, Lecturer at California State University, Northridge (2016-present)

Answered just now

byte level parity was regularly used back in the 1980's for all RAM. the 2 choices are even or odd, but mostly even parity was/is used. that means that the 9th bit, parity, is selected to make the no. of 1's always even; i.e., a 1 is applied if the no. of 1's is odd, else a 0. a "parity" checker is a logic circuit that checks for a single-bit error (or any odd number of errors):
error = XOR of all 9 bits.

Error Handling (Bit)

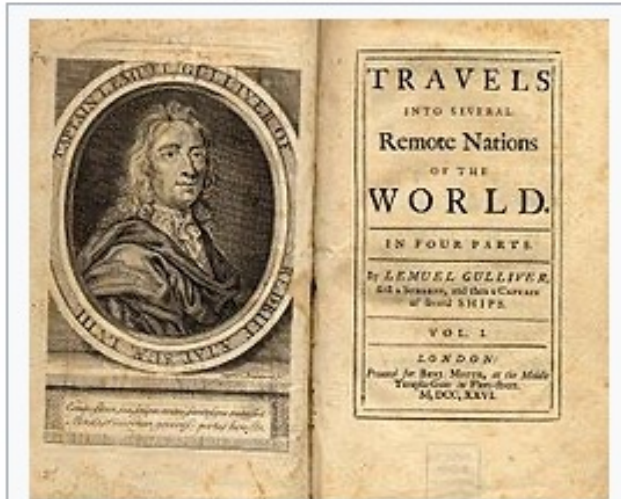
- ❖ EDC (detection)
 - ☒ Parity (byte)
 - ☐ EDC
- ❖ ECC (correction)
 - ☐ Hamming Codes

Memory Architecture

Endianness

Endianness

Gulliver's Travels



First edition of *Gulliver's Travels*

Author [Jonathan Swift](#)

Part I: A Voyage to Lilliput [\[edit \]](#)

The travel begins with a short preamble in which [Lemuel Gulliver](#) gives a brief outline of his life and history before his voyages.

4 May 1699 – 13 April 1702

During his first voyage, Gulliver is washed ashore after a shipwreck and finds himself a prisoner of a race of tiny people, less than 6 inches (15 cm) tall, who are inhabitants of the island country of [Lilliput](#). After giving assurances of his good behaviour, he is given a residence in Lilliput and becomes a favourite of the Lilliput [Royal Court](#). He is also given permission by the King of Lilliput to go around the city on condition that he must not hurt their subjects.

At first, the Lilliputians are hospitable to Gulliver, but they are also wary of the threat that his size poses to them. The Lilliputians reveal themselves to be a people who put great emphasis on trivial matters. For example, which end of an egg a person cracks becomes the basis of a deep political rift within that nation. They are a people who revel in displays of authority and performances of power. Gulliver assists the Lilliputians to subdue their neighbours the Blefuscutians by



Little Endian



Big Endian

Otherwise, if you're dealing with bytes and elements that are some multiple of bytes, reverse the order of bytes within each element. Depending your platform and language, there may be library functions to do this for you.

Fun fact: The process of converting from little endian to big endian is identical to converting from big endian to little endian, both for the egg, and for elements composed of multiple bytes.

MIPS Load Byte

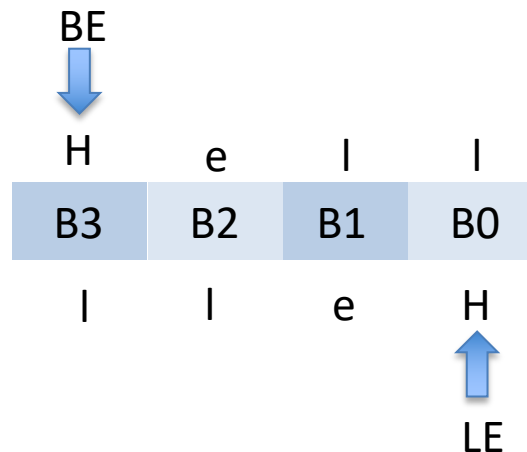
In MIPS assembly language, how does one load the address of a string's first character to a register? Should I use "load byte"?



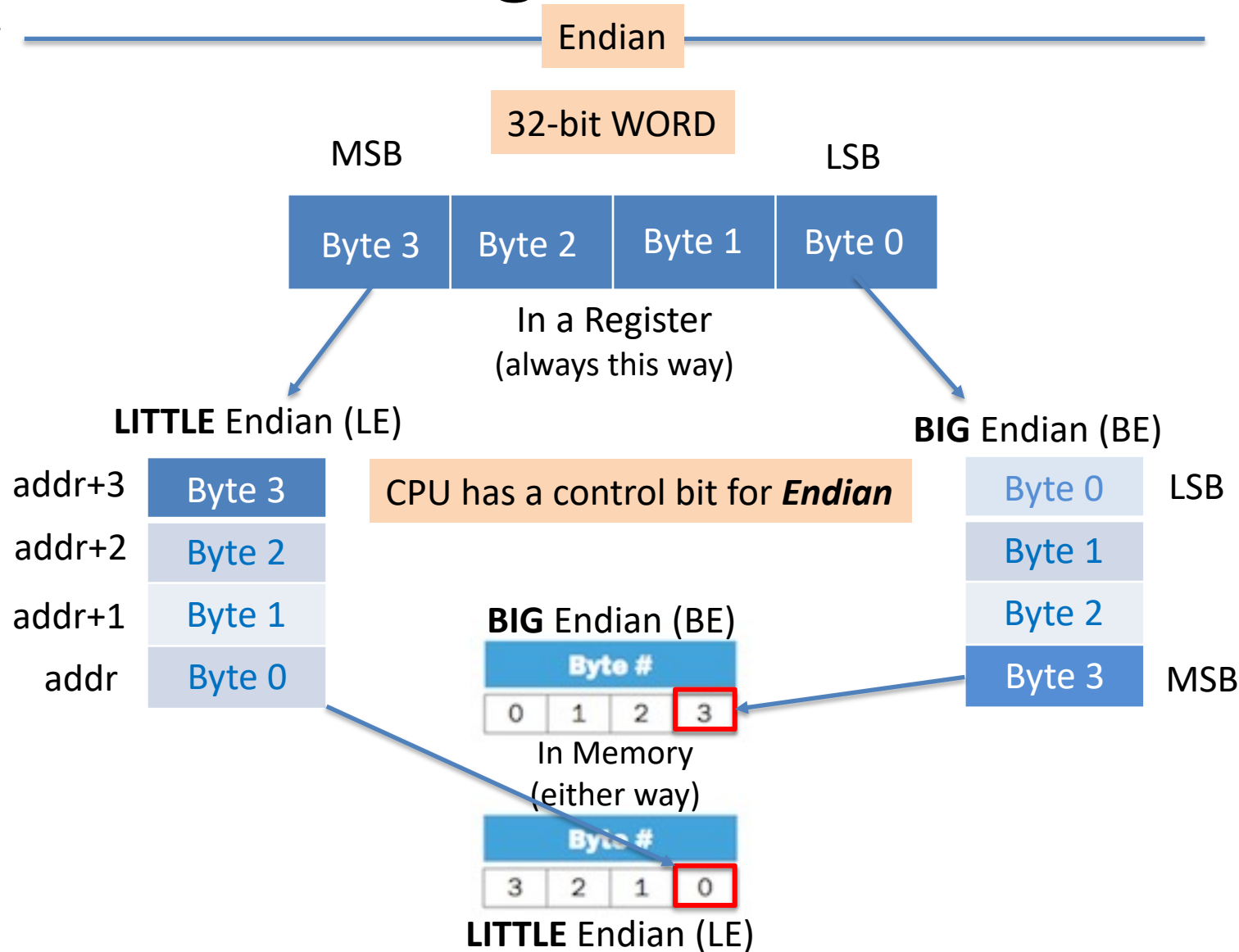
Jeff Drobman, Lecturer at California State University, Northridge (2016-present)

Answered just now

No. to load the byte of data you would use "Load byte". to load the address of the byte simply use "Load address" (a pseudo instruction) — which works for Little Endian (default). for Big endian, use "Load address" + 3.



Data: Big/Little Endian



Endianness

Endian

PARTICIPATION ACTIVITY

2.3.7: Actual MIPS memory addresses and contents of memory for those words (COD Figure 2.3).

Start

☐ 2x speed



Processor

⋮	⋮
12	100
8	10
4	101
0	1

Byte
address

Data

Memory

12	13	14	15
8	9	10	11
4	5	6	7
0	1	2	3

Showing all
byte addresses

Computers divide into those that use the address of the leftmost or "big end" byte as the word address (`_big_endian_`) versus those that use the rightmost or "little end" byte (`_little_endian_`). MIPS is in the *big-endian camp*. Since the order matters only if you access the identical data both as a word and as four bytes, few need to be aware of the endianness. (COD Appendix A (Assemblers, Linkers, and the SPIM Simulator) shows the two options to number bytes in a word.)

Memory Addresses/Contents

Endian

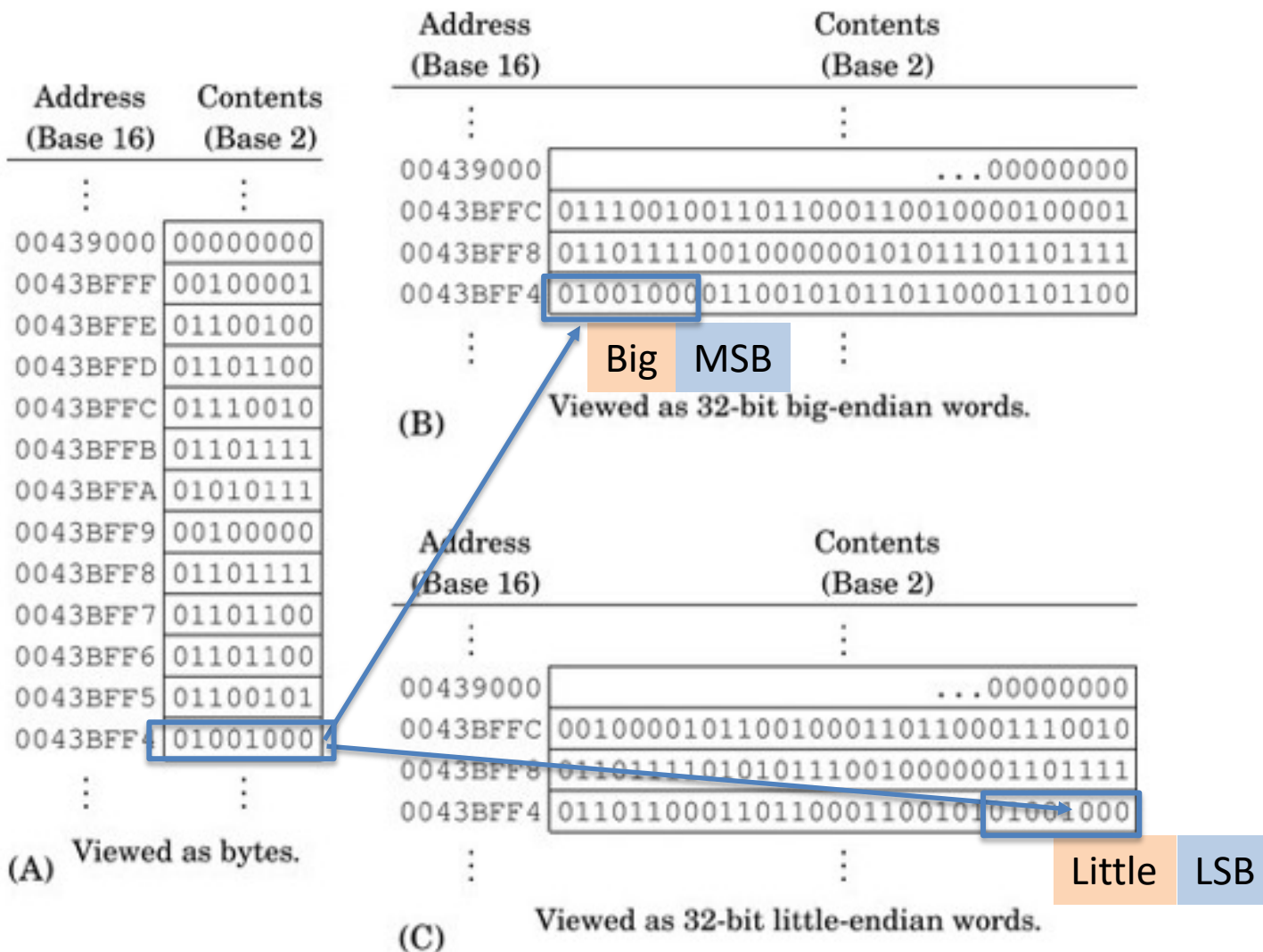


FIGURE 1.6 A section of memory.

Memory Architecture

Address Space

Ordinals

COMP122

Powers of 2 <> 10: 10:3

Technical ordinals

$10^{(-24)}$	yacto
$10^{(-21)}$	zepto
$10^{(-18)}$	atto
$10^{(-15)}$	femto
$10^{(-12)}$	pico
$10^{(-9)}$	nano
$10^{(-6)}$	micro
$10^{(-3)}$	milli
$10^{(-2)}$	centi
$10^{(-1)}$	deci
$10^{(+1)}$	deka
$10^{(+2)}$	hecto
$10^{(+3)}/2^{(10)}$	kilo
$10^{(+6)}/2^{(20)}$	mega
$10^{(+9)}/2^{(30)}$	giga
$10^{(+12)}/2^{(40)}$	tera
$10^{(+15)}/2^{(50)}$	peta
$10^{(+18)}/2^{(60)}$	exa
$10^{(+21)}/2^{(70)}$	zetta
$10^{(+24)}/2^{(80)}$	yotta

Gazillions

$10^{(+6)}$	million
$10^{(+9)}$	billion
$10^{(+12)}$	trillion
$10^{(+15)}$	quadrillion
$10^{(+18)}$	quintillion
$10^{(+21)}$	sexillion
$10^{(+24)}$	septillion
$10^{(+27)}$	octillion
$10^{(+30)}$	nonillion
$10^{(+33)}$	decillion
$10^{(+36)}$	undecillion
$10^{(+39)}$	duodecillion
$10^{(+42)}$	tredecillion
$10^{(+45)}$	quattuordecillion
$10^{(+48)}$	quindecillion
$10^{(+51)}$	sexdecillion
$10^{(+54)}$	septendecillion
$10^{(+57)}$	octodecillion
$10^{(+60)}$	novemdecillion
$10^{(+63)}$	vigintillion
$10^{(+100)}$	googol
$10^{(+303)}$	centillion
$10^{(10^{(+100)})}$	googolplex

Ordinal	Power of 2	Power of 10	Actual
1K	2^{10}	10^3	1024
1M	2^{20}	10^6	1,048,576
1G	2^{30}	10^9	1.074×10^9
1T	2^{40}	10^{12}	1.0995×10^{12}

Name	2^n	M/G	Actual
byte	2^8	--	256
short	2^{16}	64K	65,536
integer	2^{32}	4B	4.3×10^9
long	2^{64}	16 Q	1.84×10^{19}
IPv6	2^{128}	340 uD	3.4×10^{38}

GiB/TiB ($2^{30}/2^{40}$)

Decimal	Abbreviation	Value	Binary term	Abbreviation	Value	% Larger
kilobyte	KB	10^3	kibibyte	KiB	2^{10}	2%
megabyte	MB	10^6	mebibyte	MiB	2^{20}	5%
gigabyte	GB	10^9	gibibyte	GiB	2^{30}	7%
terabyte	TB	10^{12}	tebibyte	TiB	2^{40}	10%
petabyte	PB	10^{15}	pebibyte	PiB	2^{50}	13%
exabyte	EB	10^{18}	exbibyte	EiB	2^{60}	15%
zettabyte	ZB	10^{21}	zebibyte	ZiB	2^{70}	18%
yottabyte	YB	10^{24}	yobibyte	YiB	2^{80}	

Actual
1024
1,048,576
1.074×10^9
1.0995×10^{12}

Ordinal	Power of 2	Power of 10	Actual
1K	2^{10}	10^3	1024
1M	2^{20}	10^6	1,048,576
1G	2^{30}	10^9	1.074×10^9
1T	2^{40}	10^{12}	1.0995×10^{12}

Memory Architecture

Segments

MARS

MARS (MIPS Assembler and Runtime Simulator)

Memory Map

```

1  #MIPS std default memory map
2  .eqv text_seg 0x00400000
3  .eqv data_seg 0x10010000
4  .eqv heap_seg 0x10040000
5  .eqv stack_seg 0x7fffffff
6  .eqv ktext_seg 0x80000000
7  .eqv exc_ptr 0x80000180
8  .eqv kdata_seg 0x90000000
9  .eqv MMIO_seg 0xffff0000
10 .eqv memtop_ptr 0xffffffff
11 #end map

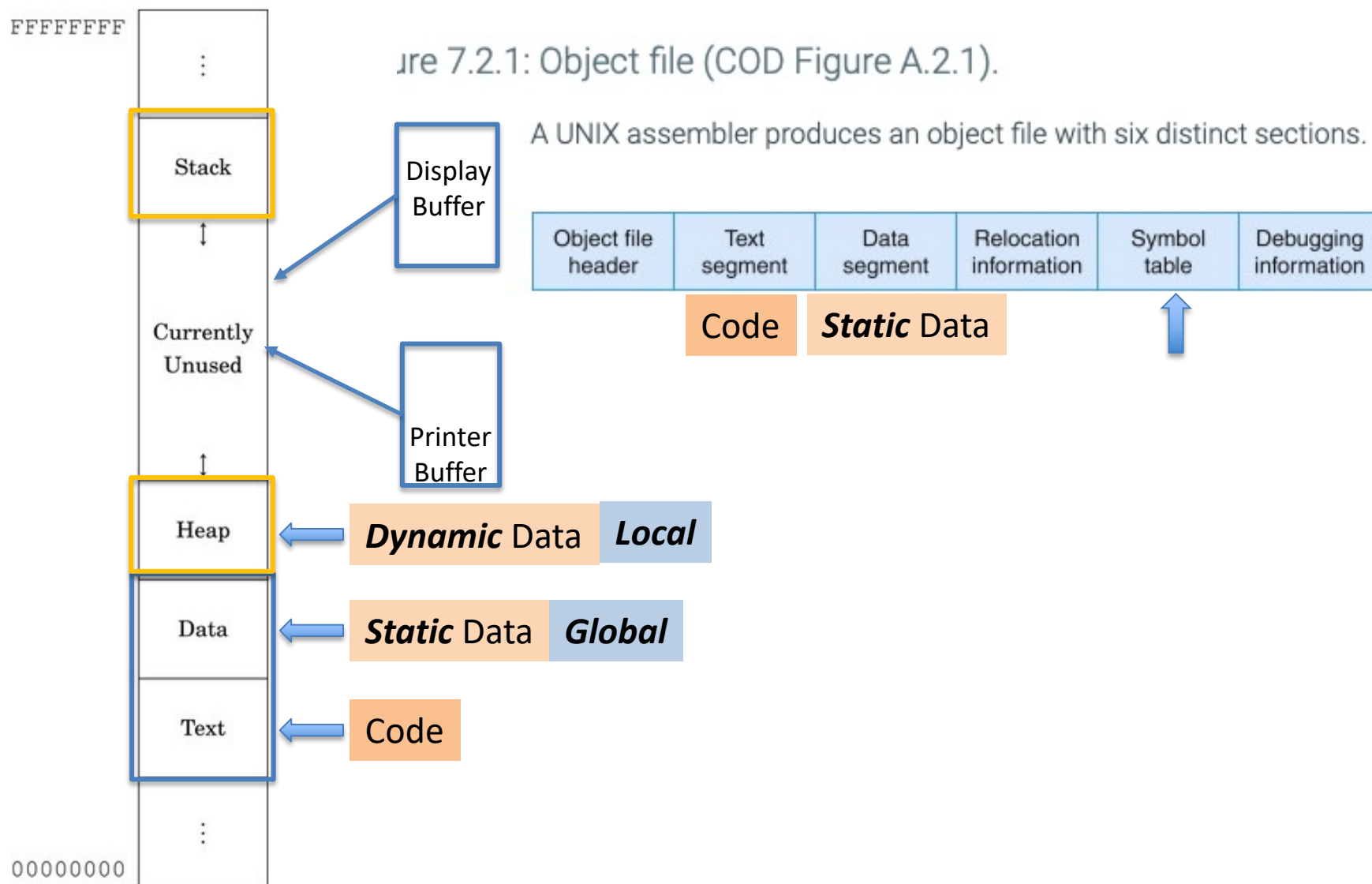
```

- ☒ Default
☐ Compact, Data at Address 0
☐ Compact, Text at Address 0

MIPS Memory Configuration

0xffffffff	memory map limit address
0xffffffff	kernel space high address
0xffff0000	MMIO base address
0xffffefff	kernel data segment limit address
0x90000000	.kdata base address
0x8fffffff	kernel text limit address
0x80000180	exception handler address
0x80000000	kernel space base address
0x80000000	.ktext base address
0x7fffffff	user space high address
0x7fffffff	data segment limit address
0x7fffffff	stack base address
0x7ffffeff	stack pointer \$sp
0x10040000	stack limit address
0x10040000	heap base address
0x10010000	.data base address
0x10008000	global pointer \$gp
0x10000000	data segment base address
0x10000000	.extern base address
0x0ffffeff	text limit address
0x00400000	.text base address

Memory Segments



Memory Segments

PARTICIPATION ACTIVITY

10.3.1: Use of the four memory regions.

Start

☐ 2x speed

```
// Program is stored in code memory
public class MemoryRegionEx {
    public static int myStaticField = 33;

    public static void myMethod() {
        int myLocal;           // On stack
        myLocal = 999;
        System.out.print(" " + myLocal);
    }

    public static void main(String[] args) {
        int myInt;              // On stack
        Integer myInteger = null; // On stack
        myInt = 555;

        myInteger = new Integer(222); // In heap
        System.out.print(myInteger.intValue() +
                        " " + myInt);

        myInteger = null;

        myMethod(); // Stack grows, then shrinks
    } // Object deallocated automatically
}
```

Code memory

1	Add R1, #1, R2
2	Sub R3, #1, R4
3	Add R1, R3, R5
4	Jmp 40

... Static memory

3000	33	myStaticField
3001		

... Stack

3200	555	myInt	main()
3201	null	myInteger	
3202	999	myLocal	
3203			

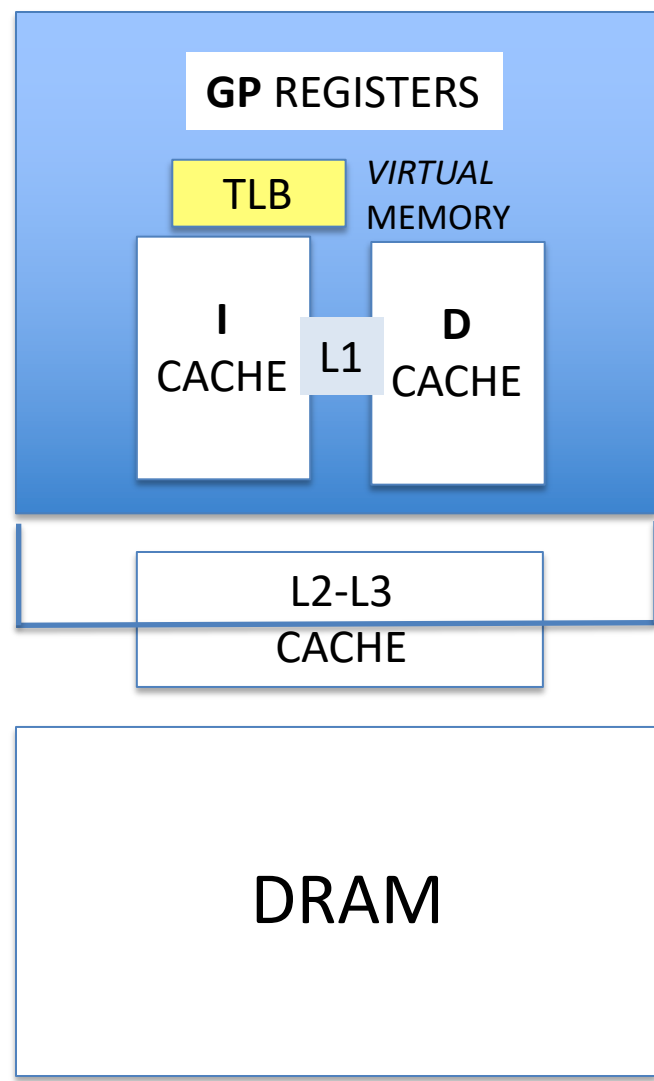
MyMethod()

... Heap

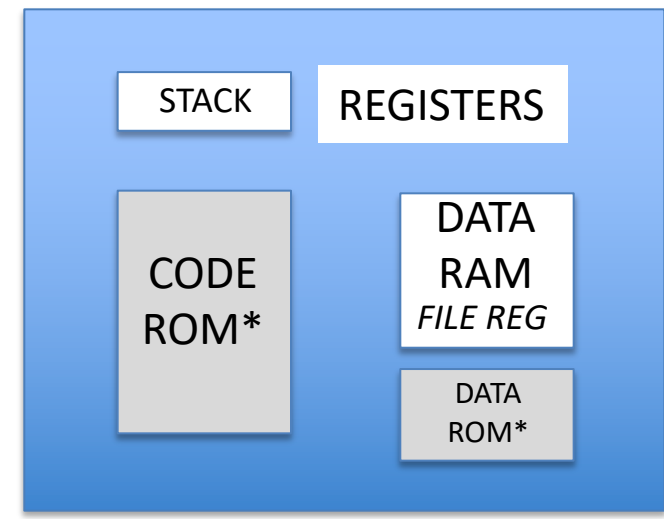
9400	222	Integer object
9401		
9402		

Memory: CPU vs MCU

MICROPROCESSOR



MICROCONTROLLER



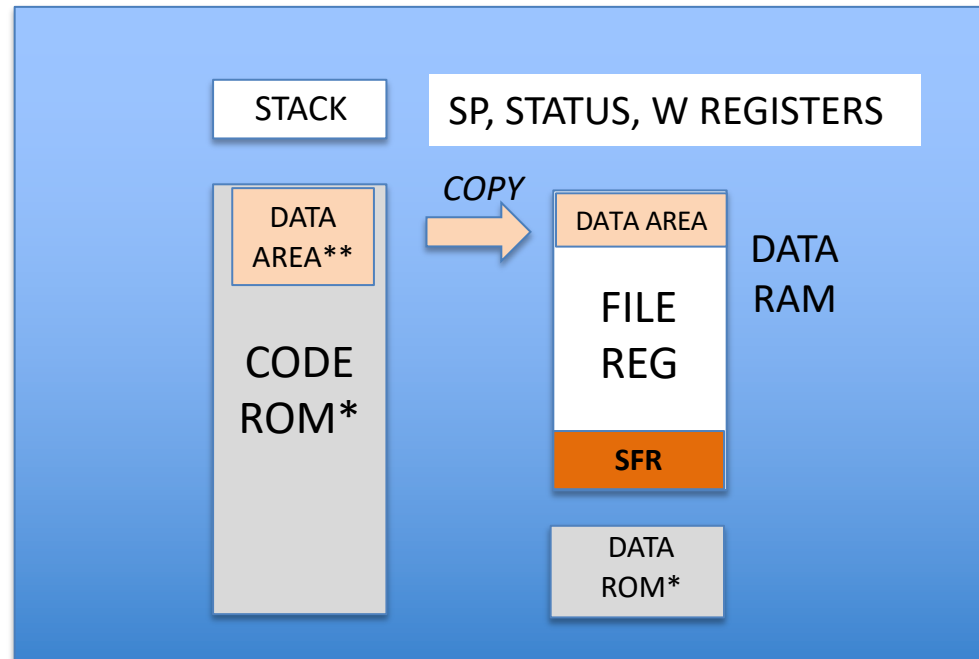
SMALL
INTERNAL
MEMORY

**ROM contents must be "programmed" "burned", or masked*

LARGE
EXTERNAL
MEMORY

PIC18F Dual Memory

PIC 18F MICROCONTROLLER



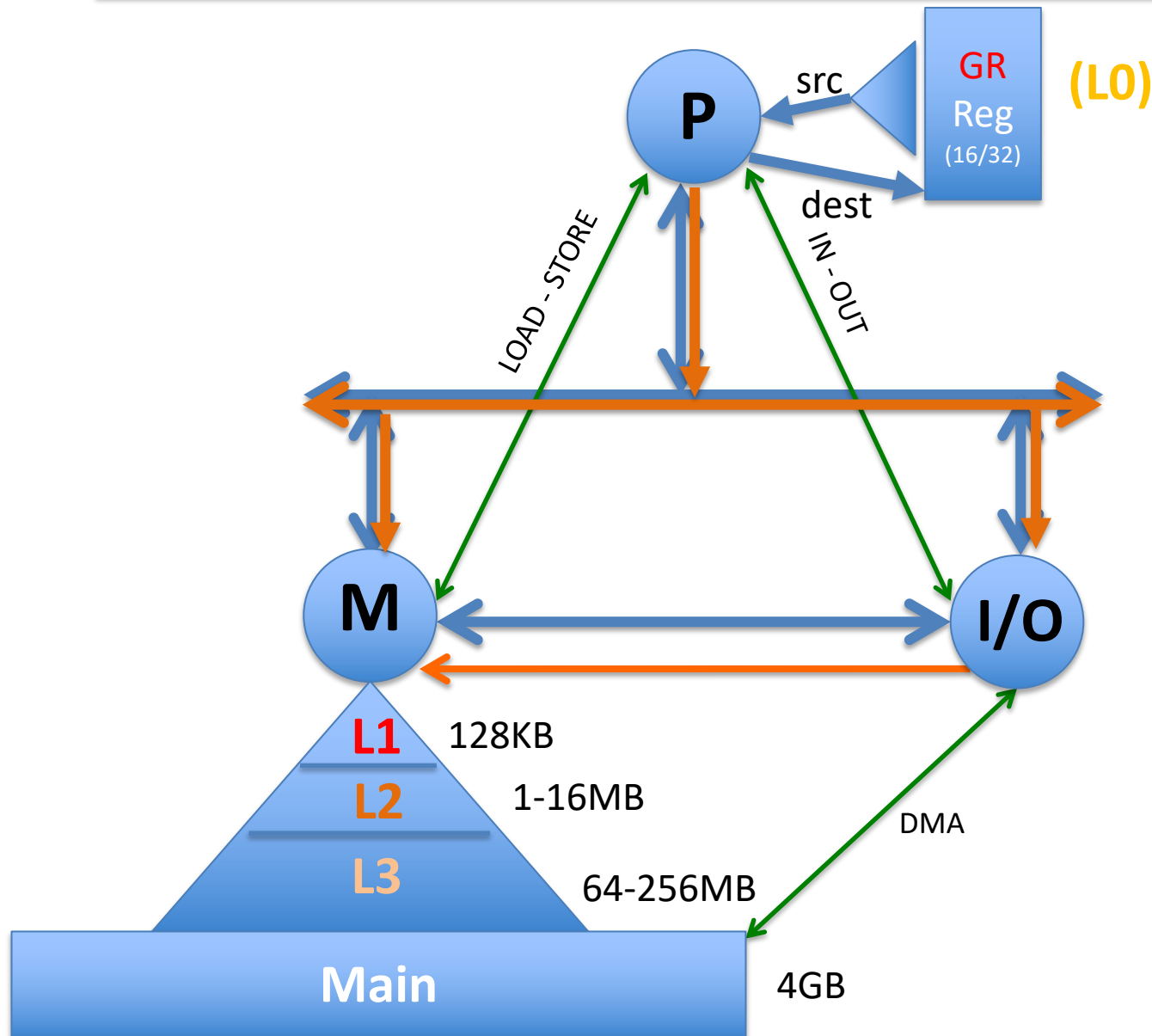
*READ ONLY

use **DB, DW

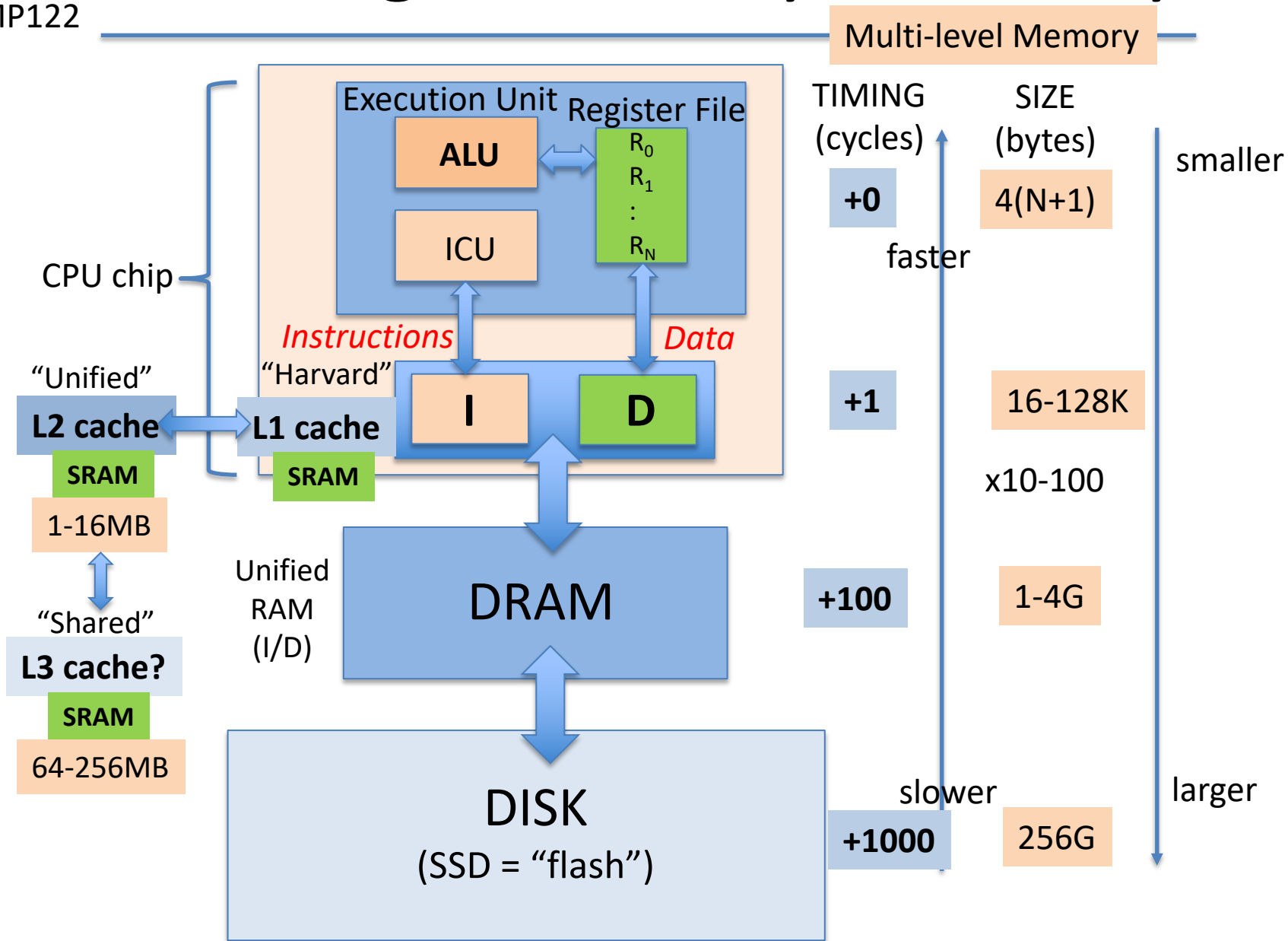
Memory

Cache

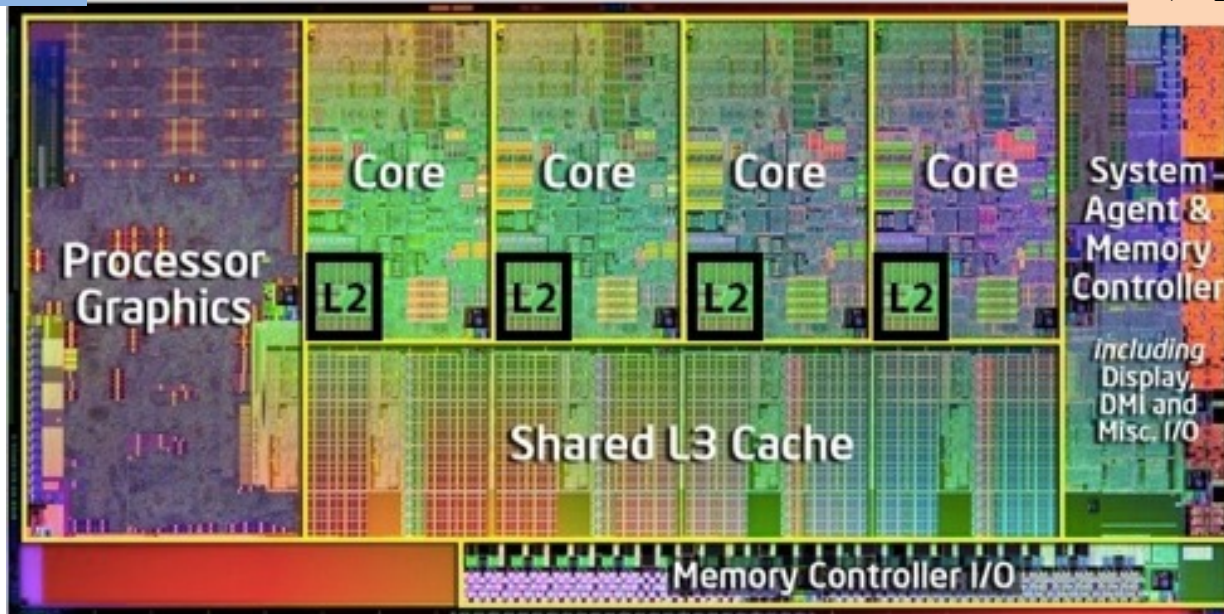
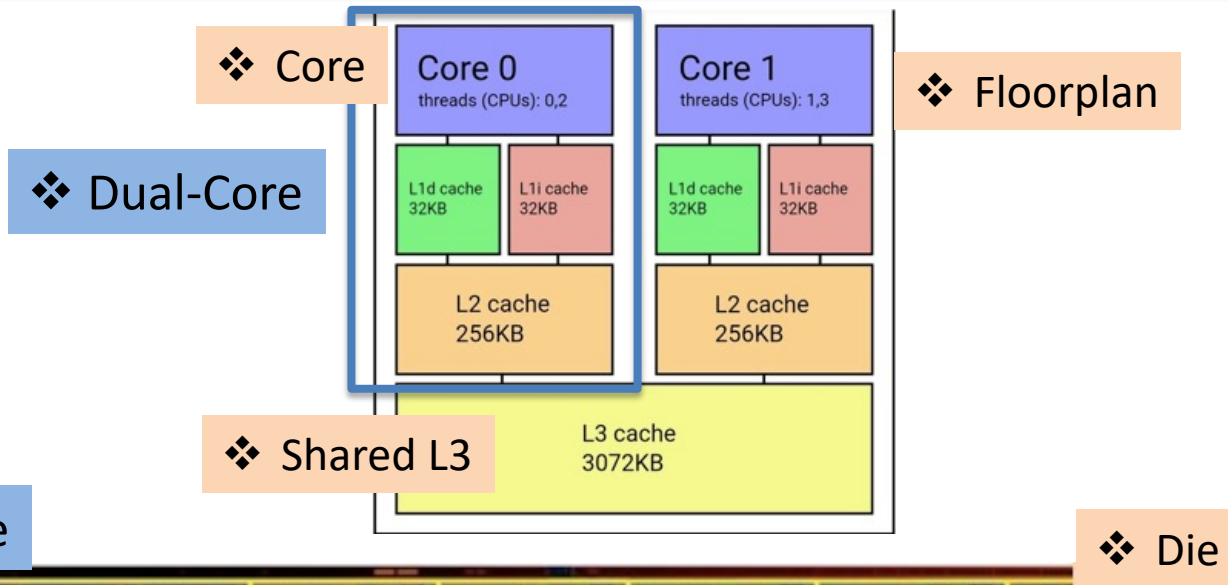
System Org: Multilevel Memory



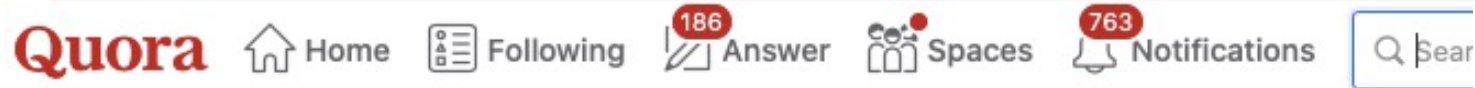
CPU Org + Memory Hierarchy



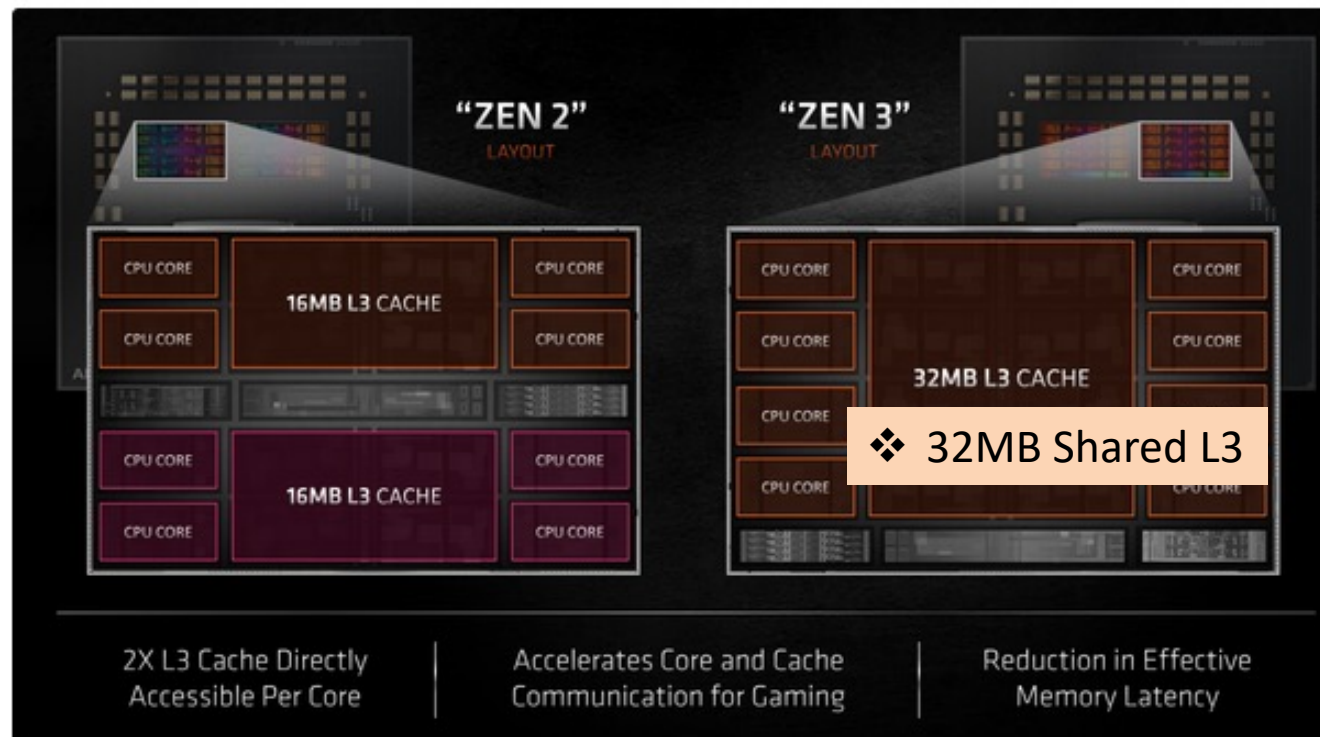
3 Levels of Cache



AMD Zen L3 Cache



One area that AMD has lagged behind Intel over the lifetime of the Zen brand is in gaming performance. It's no secret that in the company's push to lower the cost per core of its flagship processors (through the introduction of chiplet-based architectures), the design decisions have resulted in more latency between core complexes. That manifests itself in reduced performance in certain PC gaming scenarios--especially at the favored 1080p resolution used by most gamers.



This is down to how chips are designed, and, more specifically, how they're laid out on

Level 3 (LLC) Cache Size

64-256 MB → 16-64 MB



Zaky Hassani · September 21, 2019

"L3 caches tend to be in the 100s of MBs"

No, not that much. Intel's fastest CONSUMER desktop CPU, I7 9900K, has only 16MB of "intel smart cache". AMD's fastest consumer desktop CPU, the upcoming R9 3950X has 64MB of L3 cache.

What is the technical difference between a 4MB cache and 4MB L3 cache?



Jeff Drobman, Lecturer at California State University, Northridge (2016-present)

Answered just now

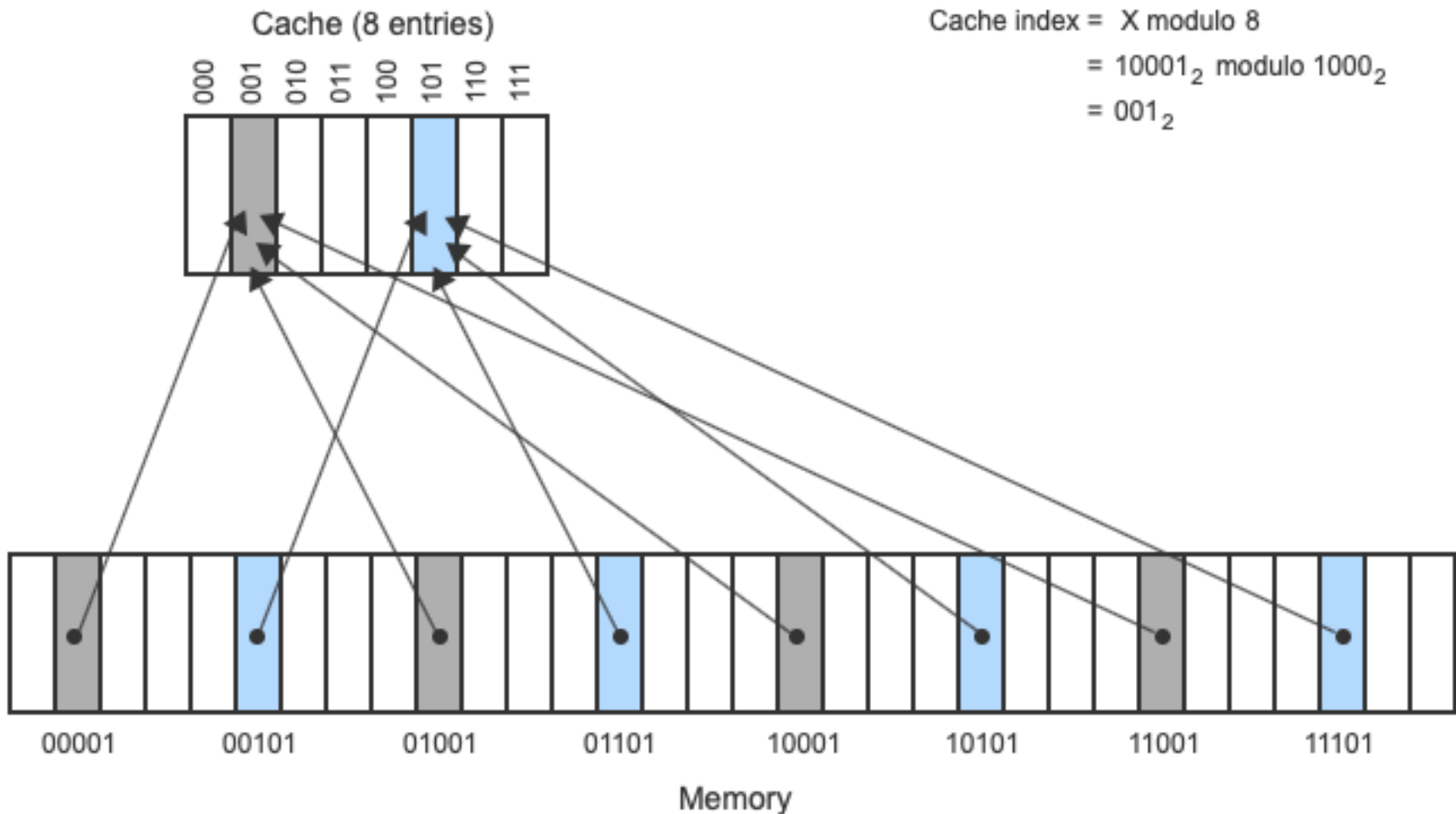
there are 3 hierarchical levels of cache memory: L1, L2, L3. L1 is for immediate access of separate I and D (**Harvard** style) operating at the CPU speed. L2 is backup larger **unified** cache per CPU core, and about 5–10x slower. L3 if used at all, is a larger/slower still "last-level" cache (LLC) that is **shared** among cores.

Cache Memory

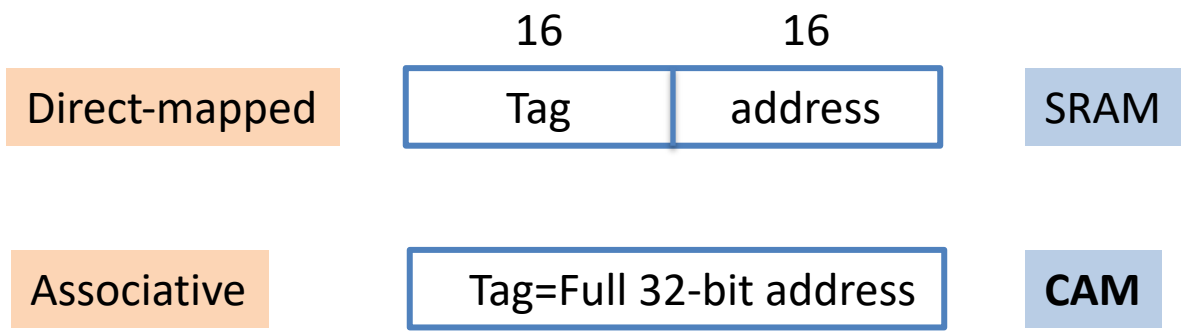
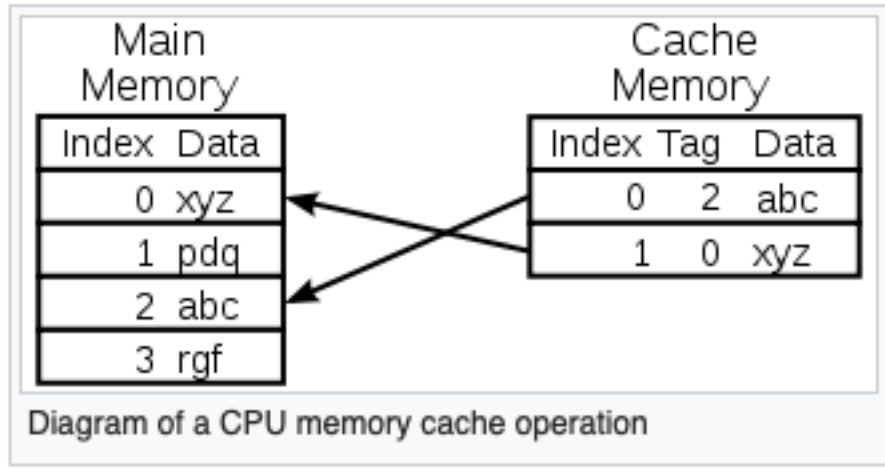
P&H Ch 5

**COMP 122: Computer
Architecture and
Assembly Language**
Spring 2020

5.3.1: A direct-mapped cache with eight entries s
map to the same cache locations (COD Figure 5.



MLM – Cache




Cache Memory

P&H Ch 5

COMP 122: Computer
Architecture and
Assembly Language
Spring 2020

Valid Bit



Index	V	Tag	Data
000	Y	10 _{two}	Memory (10000 _{two})
001	Y	11 _{two}	Memory (11001 _{two})
010	N		
011	N		
100	Y	00 _{two}	Memory (00100 _{two})
101	N		
110	N		
111	Y	10 _{two}	Memory(10111 _{two})

Invalid
data



Cache **flush** V=0

P&H Ch 5

[illegible]

L1 Cache Line Sizes

Quora



Search Q



Joe Zbiciak, I have been programming since grade school

Answered 9h ago

I & D

In the *current* crop of desktop processors, the answer is "mostly yes."

- Intel x86 CPUs since Pentium 4: yes
- AMD x86 CPUs since K7: yes
- ARM Cortex-A and Neoverse-N CPUs: yes
- PowerPC 620 (and onward?): yes
- Apple M1: **no**. [128B L1 cache line size, apparently?](#)

16 Byte 4W

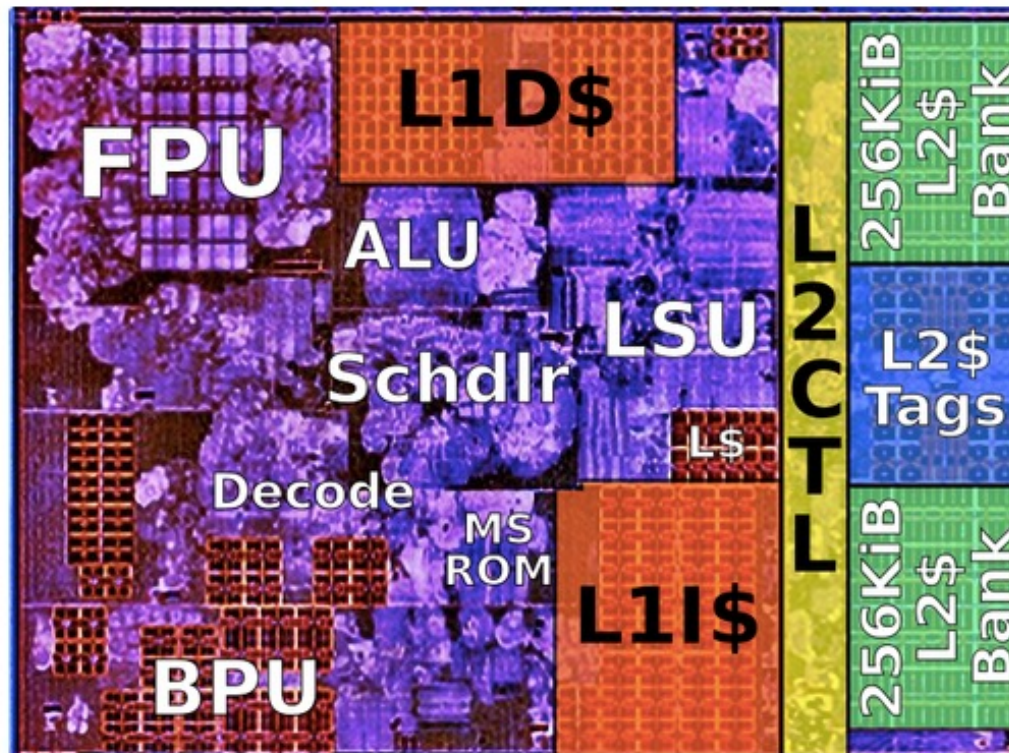
128 Byte 32W

Historically, the answer is "no."

- POWER processors use 128 byte linesize. (POWER 7 onward that I can verify; not sure about earlier POWER processors.)
- Pentium, Pentium II, Pentium III, AMD K5, AMD K6, PowerPC { 603, 603e, 604, 604e, 740, 750 } all used 32 byte L1 line sizes.
- The 80486, 68030, 68040 and 68060 used a 16 byte L1 line size.
- The 80386 and earlier x86 chips do not have on-chip cache.
- The 68020 only has an on-chip instruction cache, with 64×4 -byte entries.
- The 68000 and 68010 have no on-chip caches. The 68010 did have a small loop buffer, though.

AMD Zen Cores

To make the CPUs execute the serial code as quickly as possible, they have very big and complex cores which have things like very advanced branch predictors and huge caches to minimize time wasted for stalls. The actual execution units are only very small part of the transistor count or area of the core.



Here is a picture of AMD Zen core. The very small part "ALU" consists all the execution units for most commonly used integer instructions, and FPU is mostly the execution units of the floating point and SIMD instructions. L1D\$, L1I\$ and L2\$ are cache memory, BPU is branch prediction unit which just tries to guess what the core should do next to minimize stalls.

Intel CPU's

Power Mgt

LLC - Dynamic Cache Shrink Feature

- **LLC organized in 16 ways.**
- **When PCU detects low activity workload**
 - Flushes 14 ways of the cache and puts ways to sleep
 - Shrinks active ways from 16 to 2 to improve VccMin
- **When PCU detects high activity**
 - Expands active ways back to 16 to improve cache hit rate.

Power Down LLC

Active

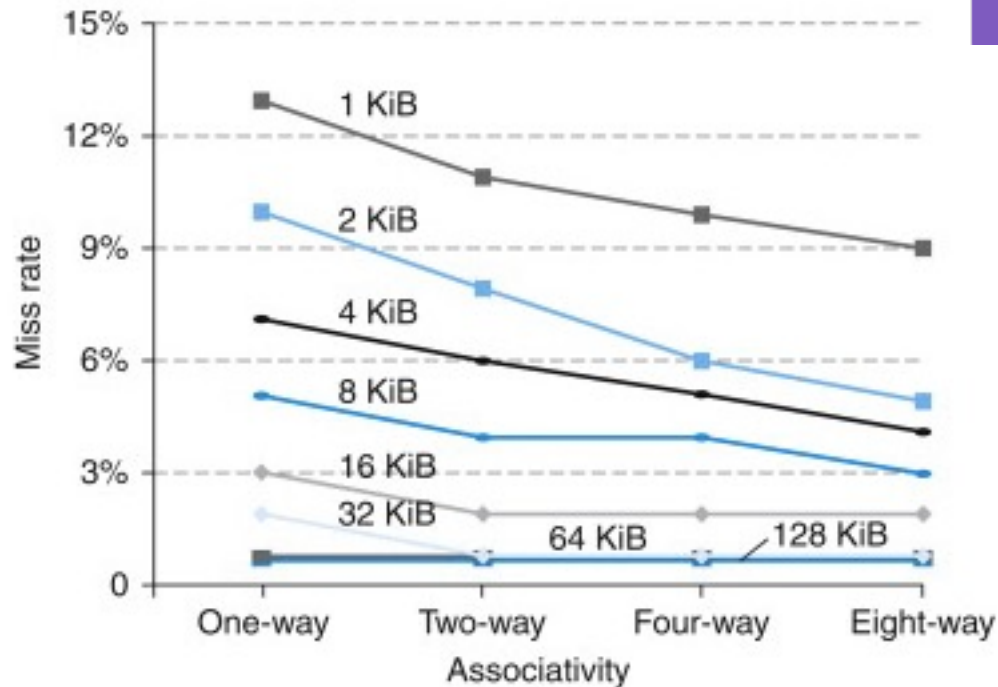
Sleep



Cache Memory

P&H Ch 5

**COMP 122: Computer
Architecture and
Assembly Language**
Spring 2020

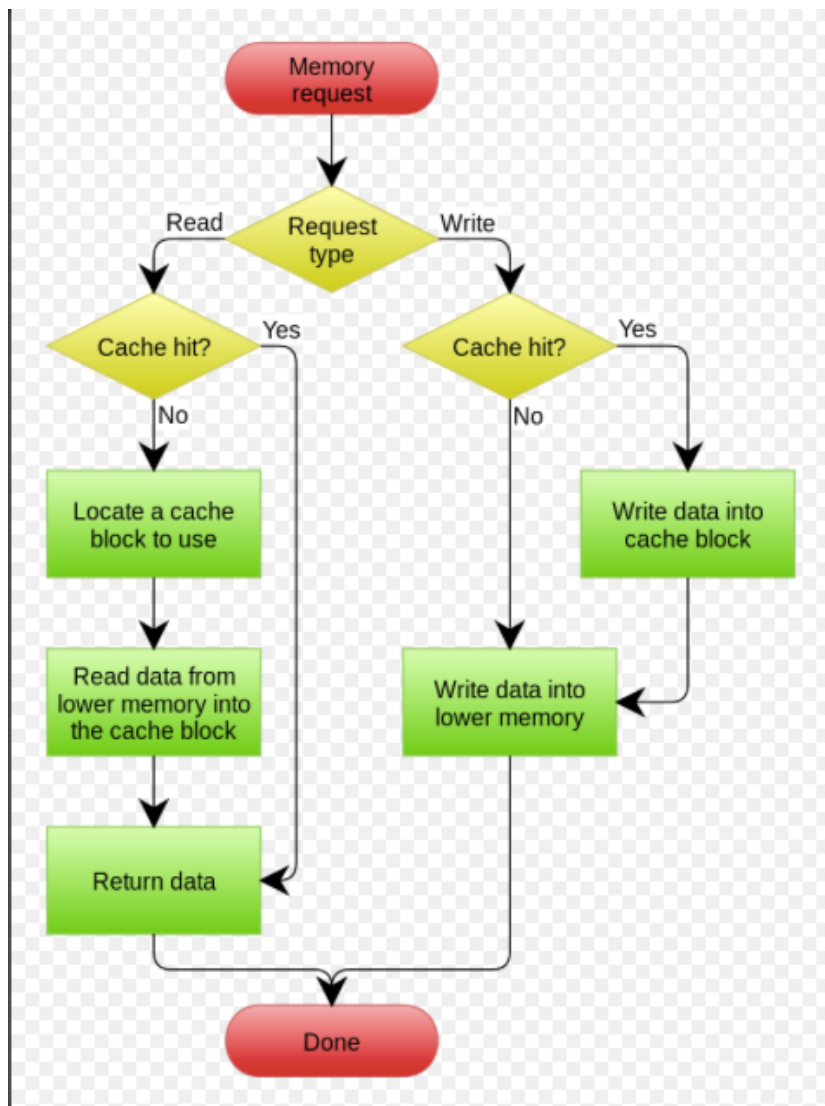


Scheme name	Number of sets	Blocks per set
Direct mapped	Number of blocks in cache	1
Set associative	$\frac{\text{Number of blocks in the cache}}{\text{Associativity}}$	Associativity (typically 2–16)
Fully associative	1	Number of blocks in the cache

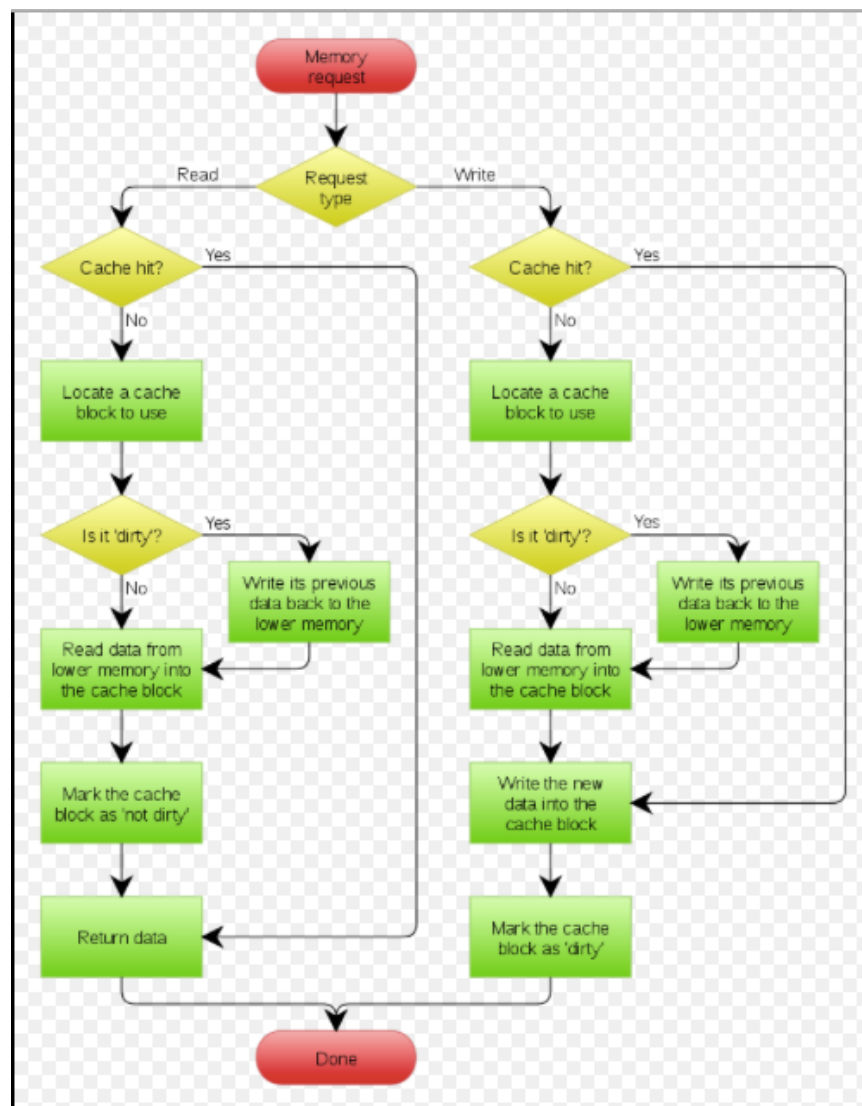
Cache Write Policies

COMP122

A write-through cache with no-write allocation



A write-back cache with write allocation



Cache Replacement



Joe Zbiciak, Developed practical algorithms actually used in production.

Hacking vulnerability

LRU vs Random replacement

In a cache with low associativity (e.g. 4 ways or fewer), the odds you evict something useful are much higher with random replacement. Also, a true random replacement policy doesn't let you put streaming data near the "least" end of an LRU.

It's not clear to me that random replacement actually fixes the vulnerability, either. It may just mean that the attacker has to collect more data before they are certain about the signal.

Cache timing attacks rely on two items mapping to the same set. Random replacement adds noise, but it does not remove signal, because it did not change how data maps to sets.

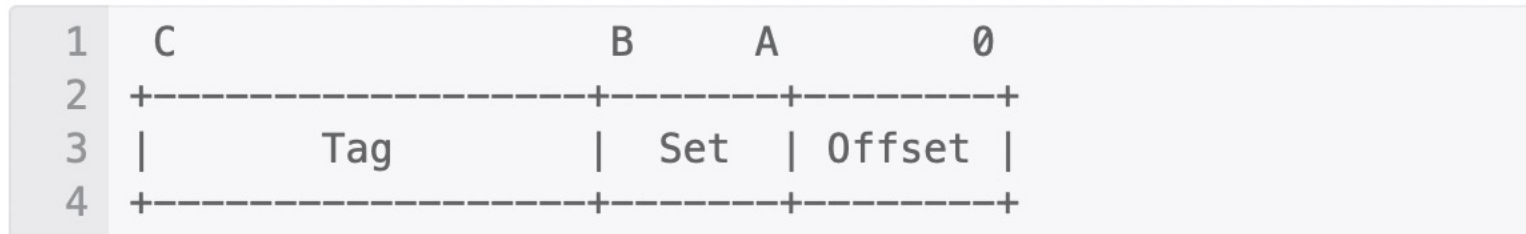
Cache Replacement



Joe Zbiciak, Developed practical algorithms actually used in production.

I was going to add a link to describe set-skewing, but I could not find a good one. What I am referring to here is using a hash function that looks at the entire address above the offset field, instead of just extracting the a contiguous set of address bits.

A traditional direct-mapped or set-associative cache extracts a fixed field of the address to determine the set:



Two addresses map to the same set if bits $[B : A]$ match.

In a set-skewed cache, you put bits $[C : A]$ into a hashing function to select a set. Two addresses map to the same set if their hashes collision.

And yes, technically speaking, the non-skewed approach is the same as the skewed approach with hash function that just extracts the lower n bits (i.e. computes modulo 2^n). For this use case you'd clearly need a more involved hashing function.

Memory

Virtual Memory

MMU/TLB

What is the definition of a page in computer science?

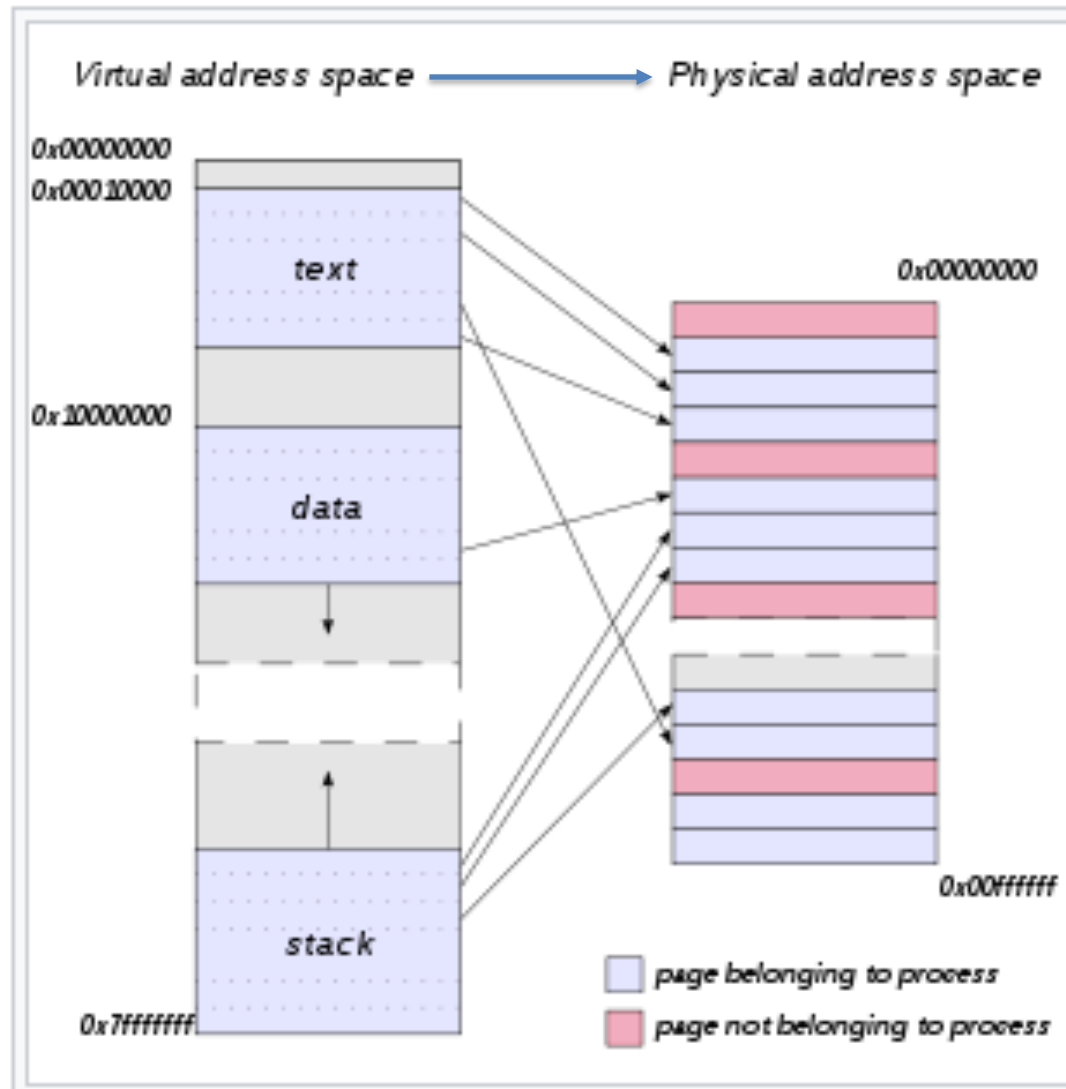


Jeff Drobman

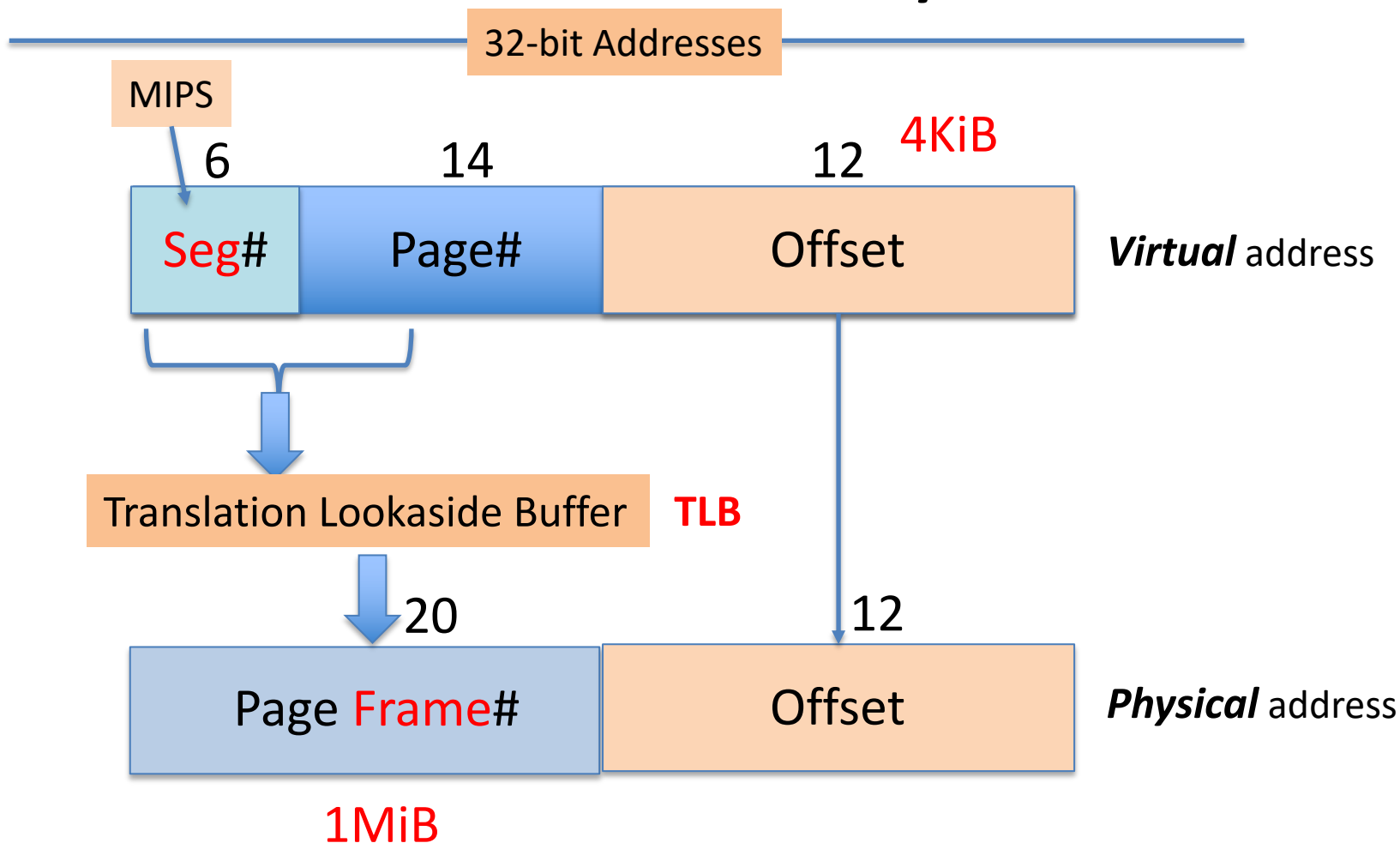
Lecturer at California State University, Northridge (2016–present) · Just now

a “page” is a unit of main memory segmentation anywhere in size from 1KB to 1MB, but typically 64KiB (e.g., MIPS). there can be both *virtual* and *physical* pages in a virtual memory, with a mapping managed by the OS. in a virtual memory, the upper part of a virtual address is the page number, while the lower portion is the untranslated address within the page.

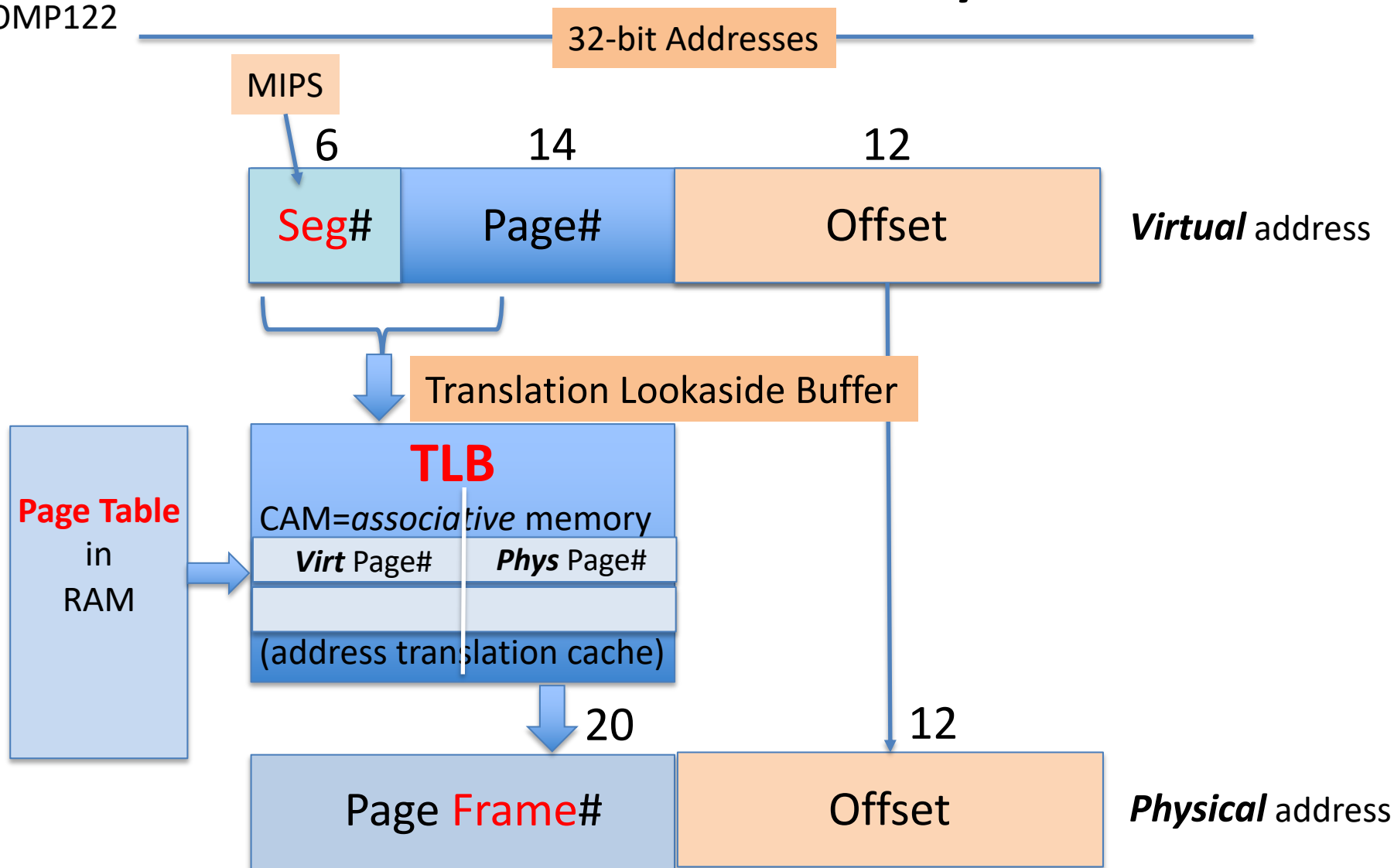
Virtual Memory



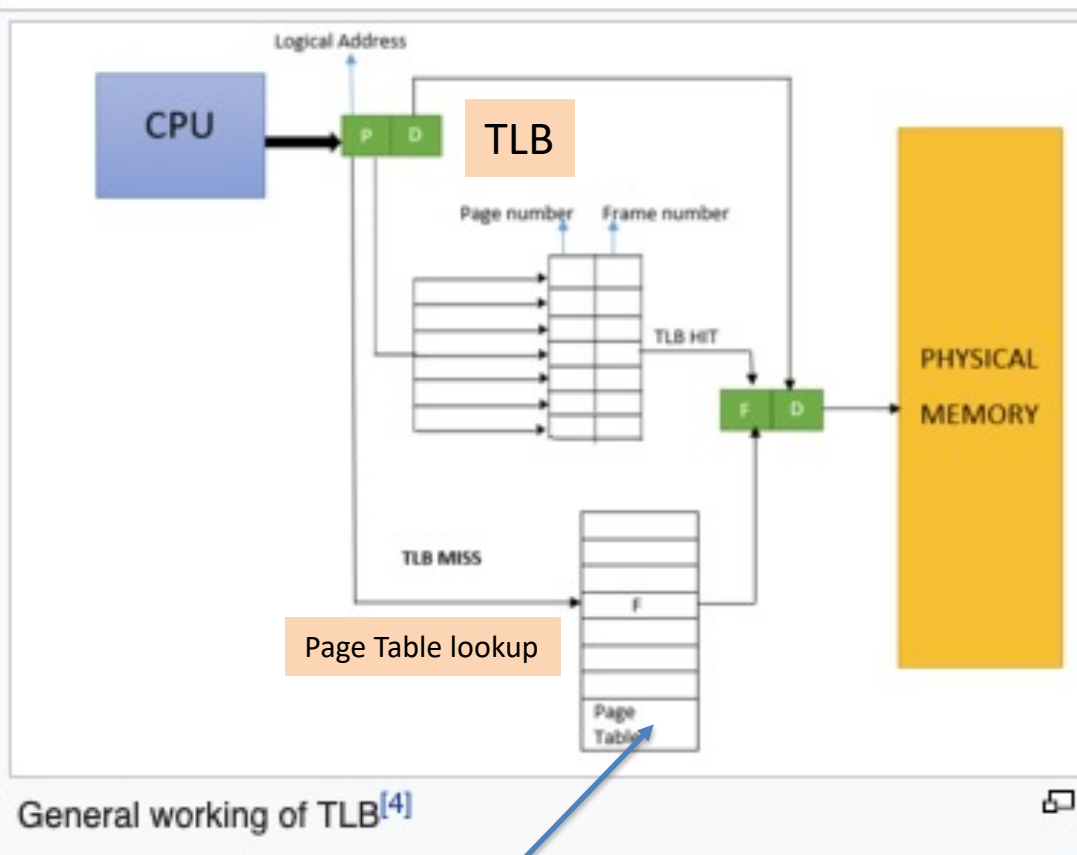
Virtual Memory



Virtual Memory



Virtual Memory

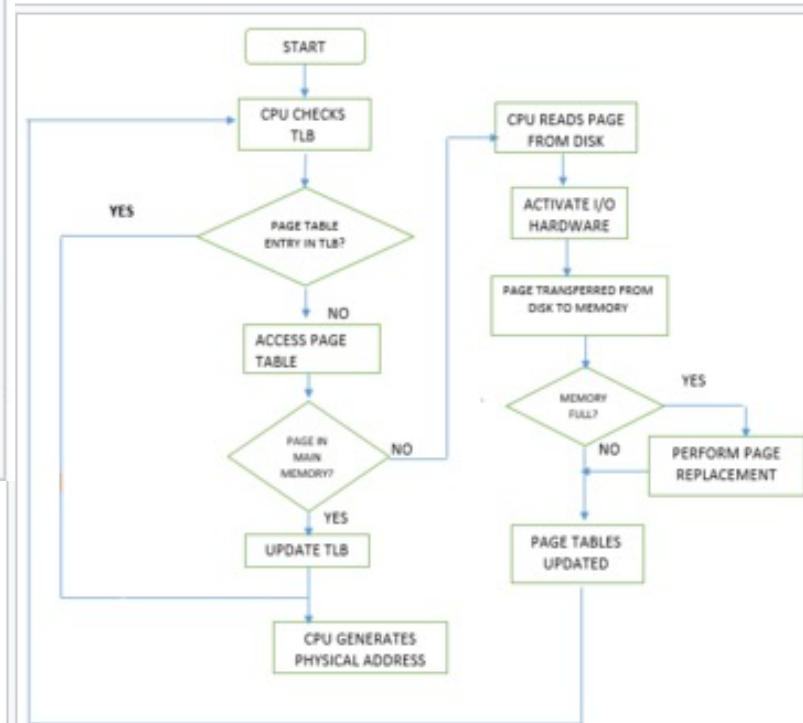


Page Table

Typical TLB [edit]

These are typical performance levels of a TLB:^[17]

- size: 12 bits – 4,096 entries
- hit time: 0.5 – 1 clock cycle
- miss penalty: 10 – 100 clock cycles
- miss rate: 0.01 – 1% (20–40% for sparse/graph applications)



Flowchart^[6] shows the working of a translation lookaside buffer. For simplicity, the page-fault routine is not mentioned.

Virtual Memory

P&H Ch 5

COMP 122: Computer
Architecture and
Assembly Language
Spring 2020

5.7 Virtual memory

Figure 5.7.2: Mapping from a virtual to a physical address

Page fault: An event that occurs when an accessed page is not present in main memory.

Virtual address: An address that corresponds to a location in virtual space and is translated by address mapping to a physical address when memory is accessed.

Address translation: Also called **address mapping**. The process by which a virtual address is mapped to an address used to access memory.

Figure 5.7.1: In virtual memory, blocks of memory (called pages) are mapped from one set of addresses (called virtual addresses) to another set (called physical addresses) (COD Figure 5.24).

Virtual Memory

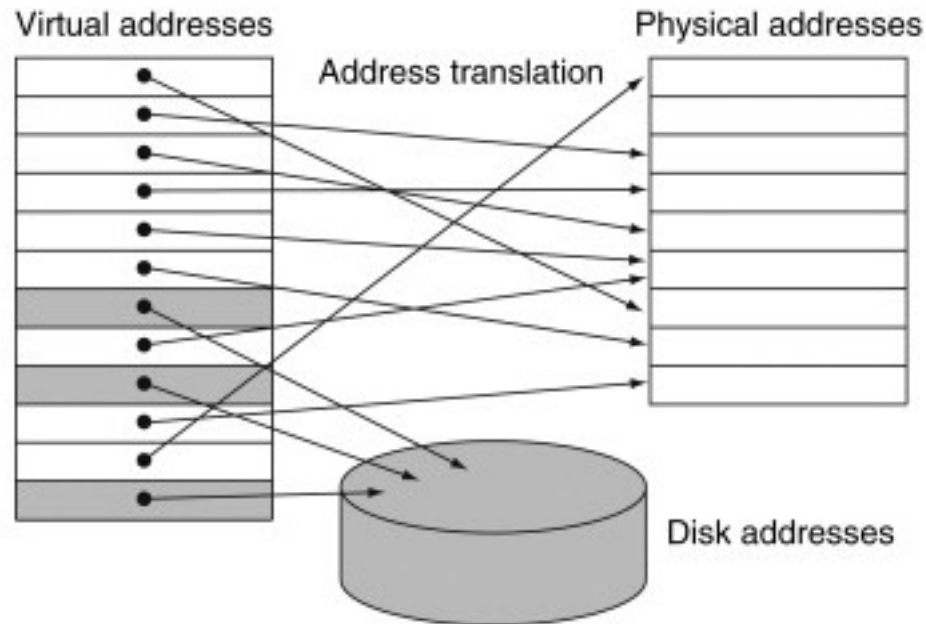
P&H Ch 5

COMP 122: Computer
Architecture and
Assembly Language
Spring 2020

5.7 Virtual memory

Figure 5.7.2: Mapping from a virtual to a physical address

Figure 5.7.1: In virtual memory, blocks of memory (called pages) are mapped from one set of addresses (called virtual addresses) to another set (called physical addresses) (COD Figure 5.24).



Virtual Memory

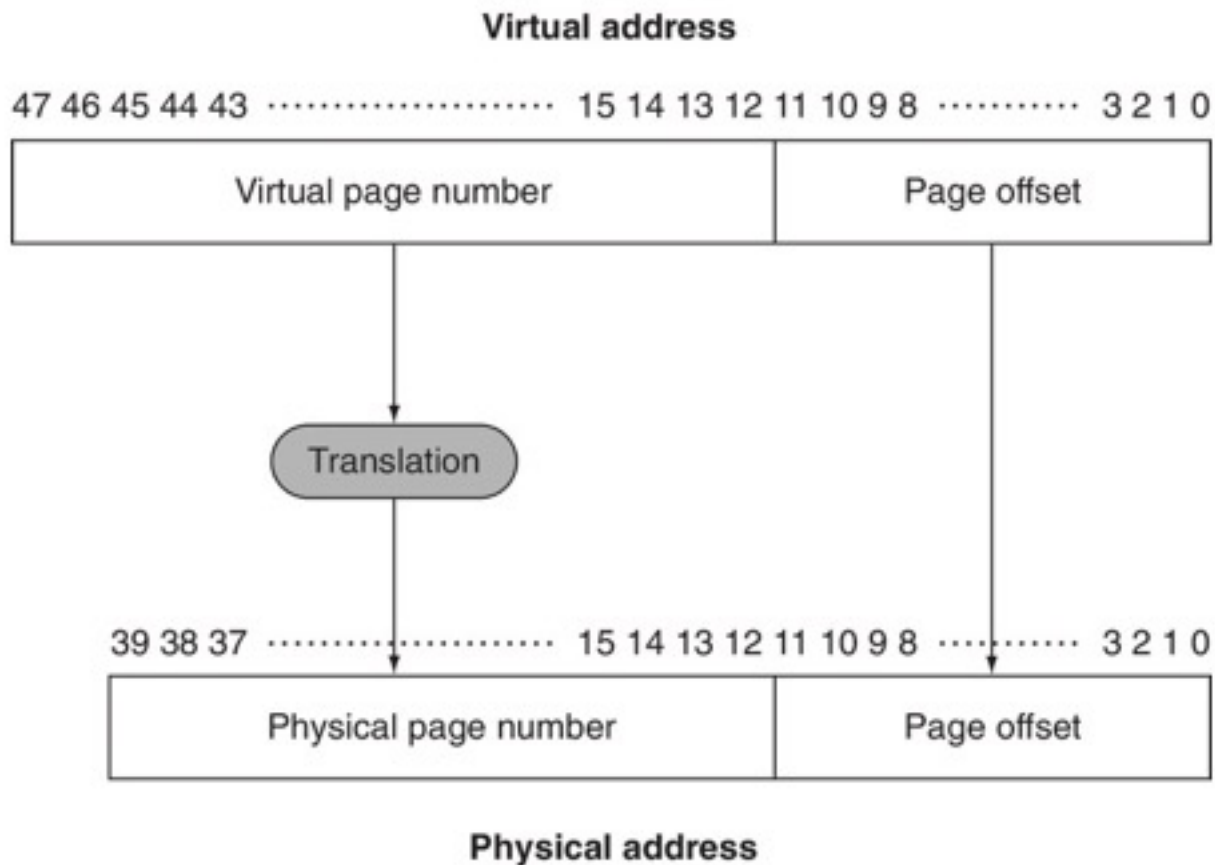
COMP122

TLB

P&H Ch 5

**COMP 122: Computer
Architecture and
Assembly Language**
Spring 2020

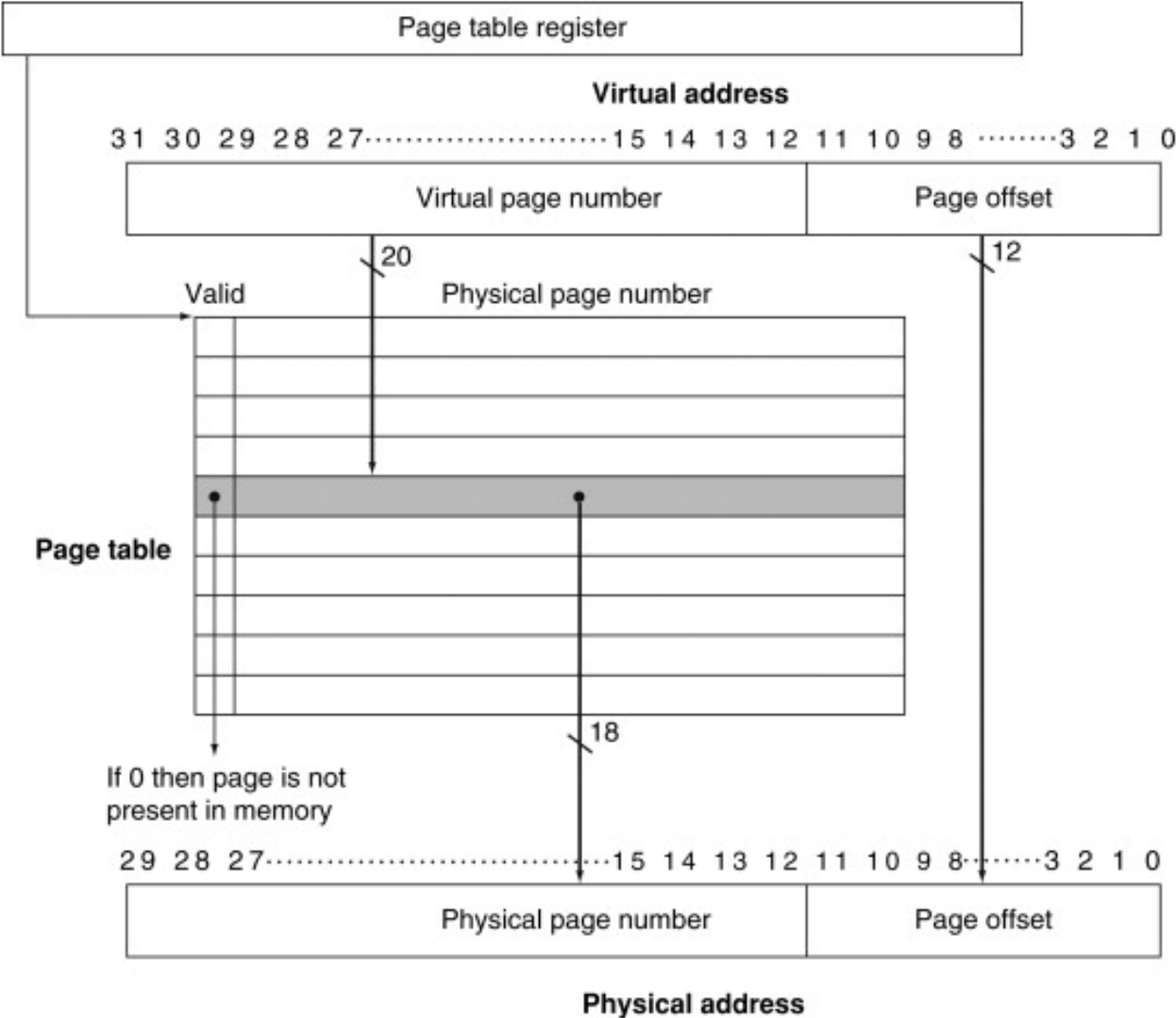
Figure 5.7.2: Mapping from a virtual to a physical address



Virtual Memory

P&H Ch 5

COMP 122: Computer
 Architecture and
 Assembly Language



MLM Sizes/Hits

P&H Ch 5

**COMP 122: Computer
Architecture and
Assembly Language**
Spring 2020

Feature	Typical values for L1 caches	Typical values for L2 caches	Typical values for paged memory	Typical values for a TLB
Total size in blocks	250–2000	2500–25,000	16,000–250,000	40–1024
Total size in kilobytes	16–64	125–2000	1,000,000–1,000,000,000	0.25–16
Block size in bytes	16–64	64–128	4000–64,000	4–32
Miss penalty in clocks	10–25	100–1000	10,000,000–100,000,000	10–1000
Miss rates (global for L2)	2%–5%	0.1%–2%	0.00001%–0.0001%	0.01%–2%

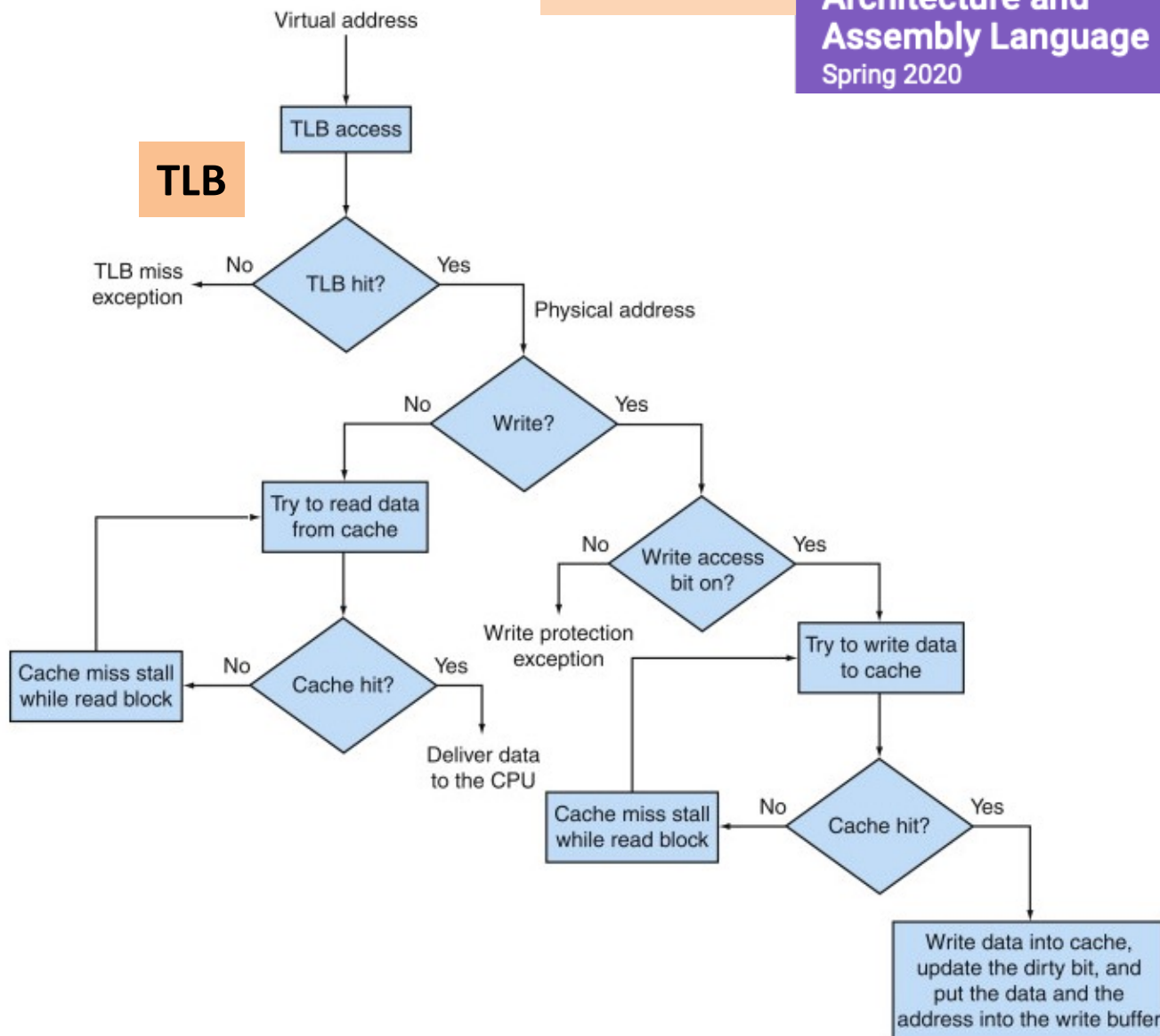
TLB

TLB	Page table	Cache	Possible? If so, under what circumstance?
Hit	Hit	Miss	Possible, although the page table is never really checked if TLB hits.
Miss	Hit	Hit	TLB misses, but entry found in page table; after retry, data is found in cache.
Miss	Hit	Miss	TLB misses, but entry found in page table; after retry, data misses in cache.
Miss	Miss	Miss	TLB misses and is followed by a page fault; after retry, data must miss in cache.
Hit	Miss	Miss	Impossible: cannot have a translation in TLB if page is not present in memory.
Hit	Miss	Hit	Impossible: cannot have a translation in TLB if page is not present in memory.
Miss	Miss	Hit	Impossible: data cannot be allowed in cache if the page is not in memory.

Virtual Memory

P&H Ch 5

COMP 122: Computer
Architecture and
Assembly Language
Spring 2020



Virtual Memory

P&H Ch 5

**COMP 122: Computer
Architecture and
Assembly Language**
Spring 2020

