DR JEFF SOFTWARE
INDIE APP DEVELOPER

COMP 222

Spring 2023

Rev 1-22-23

Computer Organization

(Architecture)

Lecture 2

Dr Jeff Drobman

website → *drjeffsoftware.com/classroom.html*

email → *jeffrey.drobman@csun.edu*

Index (Vol. 2)

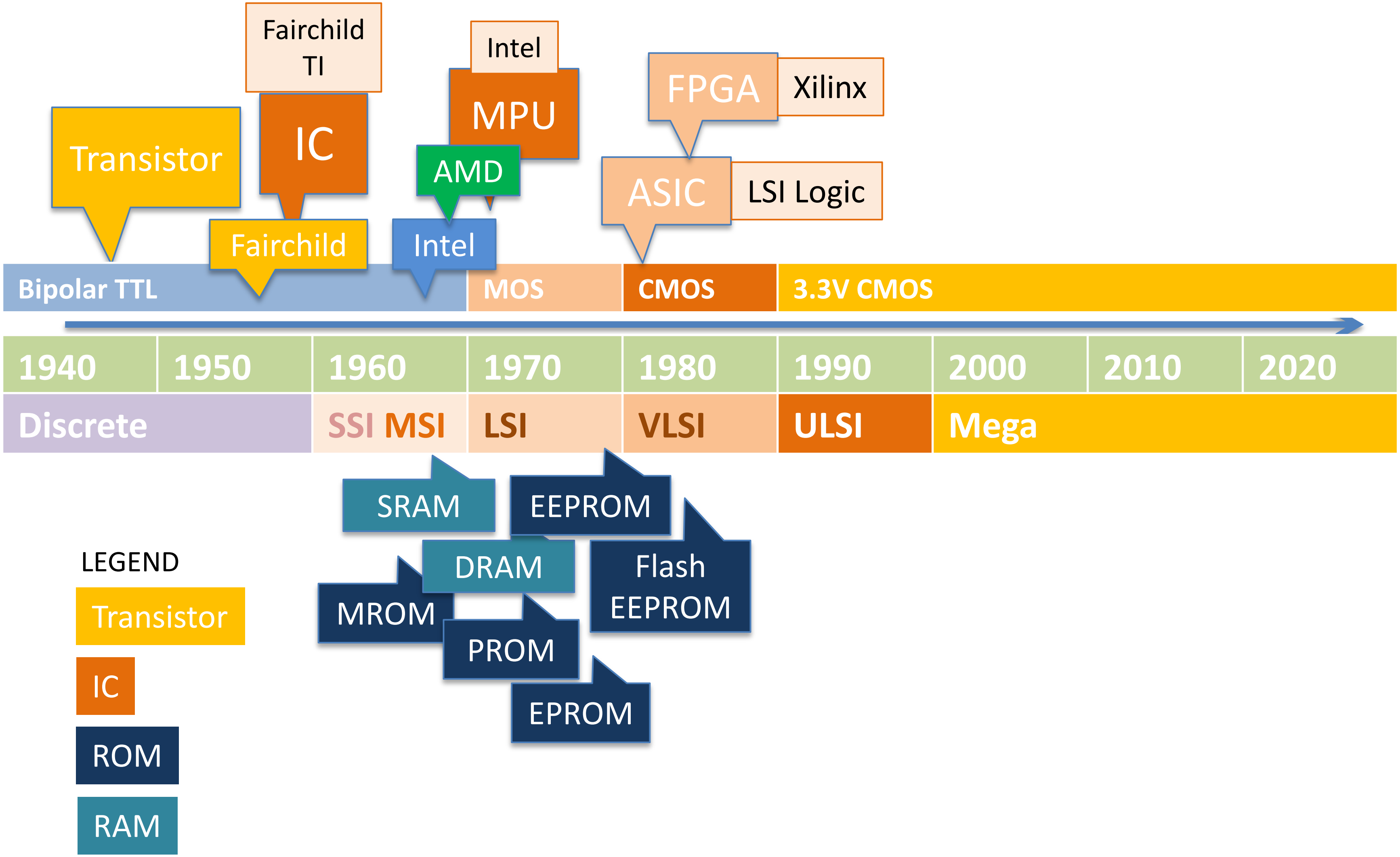
Section

Computer Chips & Microprocessor History

(see separate slide set ***CS122-Lecture 1***)

IC Technology

TIMELINE



Transistor

Fairchild
TI

IC

Fairchild

Intel

MPU

AMD

Intel

FPGA

Xilinx

ASIC

LSI Logic

1940	1950	1960	1970	1980	1990	2000	2010	2020
Bipolar TTL			MOS	CMOS	3.3V CMOS			
Discrete		SSI MSI	LSI	VLSI	ULSI	Mega		

SRAM

EEPROM

DRAM

Flash
EEPROM

MROM

PROM

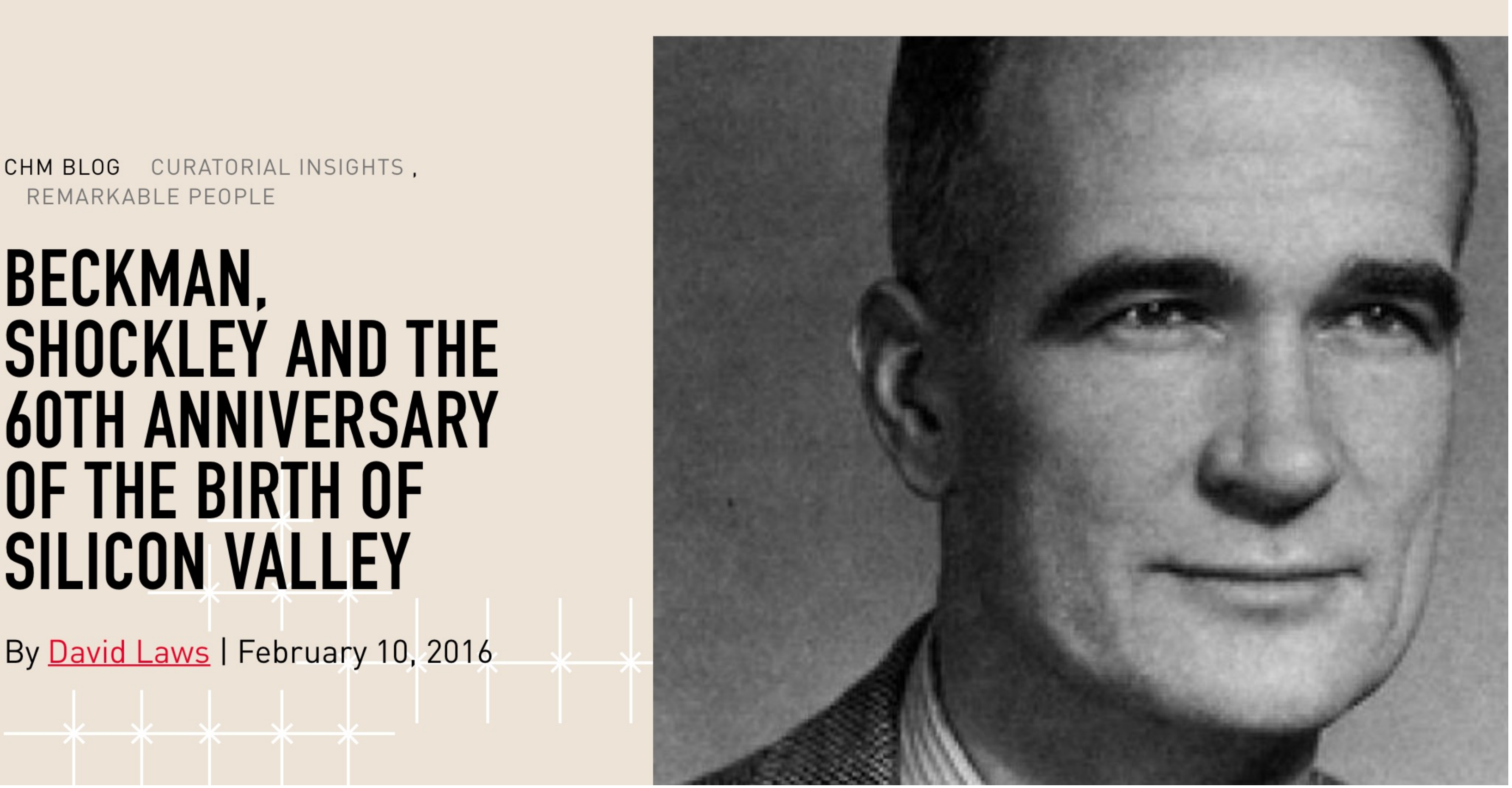
EPROM

LEGEND

- Transistor
- IC
- ROM
- RAM

CHM on Shockley

https://computerhistory.org/blog/beckman-shockley-and-the-60th-anniversary-of-the-birth-of-silicon-valley



CHM BLOG CURATORIAL INSIGHTS , REMARKABLE PEOPLE

BECKMAN, SHOCKLEY AND THE 60TH ANNIVERSARY OF THE BIRTH OF SILICON VALLEY

By David Laws | February 10, 2016

CHM on Shockley

None would have the same lasting impact on the fortunes of the future Silicon Valley and beyond as Dr. Arnold Beckman's disclosure of an agreement signed the previous day for "the establishment in the Stanford community of the Shockley Semiconductor Laboratory to develop and produce transistors and other semiconductor devices."

ABOUT THE AUTHOR

David A. Laws [AMD 1975-1986, V.P. Business Development] is a high-technology business consultant with a focus on marketing and strategic planning. He earned a B.Sc. (Physics) in the UK and after moving to California in 1968 worked for Silicon Valley companies, including Fairchild Semiconductor, Advanced Micro Devices (AMD), and Altera Corporation, in roles from product marketing engineer to CEO.

My Genesis Article

Genesis: A Silicon Valley Tale

FAIRCHILD SEMICONDUCTOR CORPORATION

TECH HISTORY ARTICLE BY DR JEFF DROBMAN

Highlights

- Fairchild founding
- Intel founding
- AMD history
- AMD – Intel rivalry
- Search for CMOS
- RISC CPU Architecture
- Legendary Parties & Conferences
- Anecdotes
- Valley Significant Others
- Genesis org-chart
- Process Technology Evolution
- Anniversaries of Technologies

SHOCKLEY SEMICONDUCTOR LABORATORY
BECKMAN INSTRUMENTS INC.
MOUNTAIN VIEW, CALIFORNIA
Shockley Semiconductor Laboratory

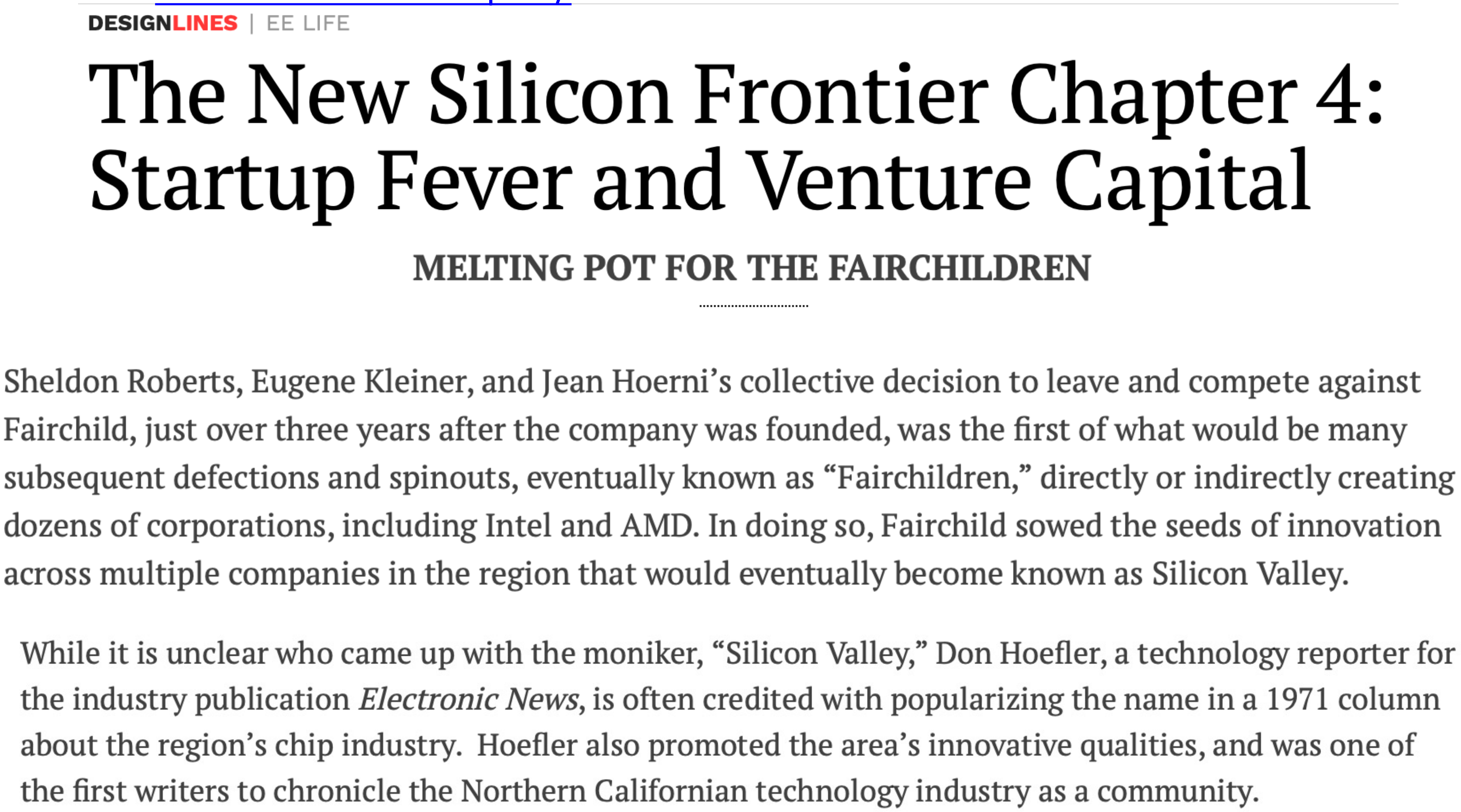
"Why don't we get together and start a business?"
Operations.
Sales and Administration.
Director of Analog Op

My Genesis Article

The Legend

It has long been *legendary* that companies in Silicon Valley got started in garages and beach houses, and I am setting the record straight: *It is true*. **Apple** was started in Steve Wozniak's garage, when friend Steve Jobs came by and saw his hobby computer. **Advanced Micro Devices** (AMD) got its start in founding president Jerry Sanders' rented Malibu beach house, on a chilly December evening in 1968 – though the house was heated considerably by those entrepreneurial fires. AMD was incorporated 5 months later (May 1969).

https://www.eetimes.com/the-new-silicon-frontier-chapter-4-startup-fever-and-venture-capital/



DESIGNLINES | EE LIFE

The New Silicon Frontier Chapter 4: Startup Fever and Venture Capital

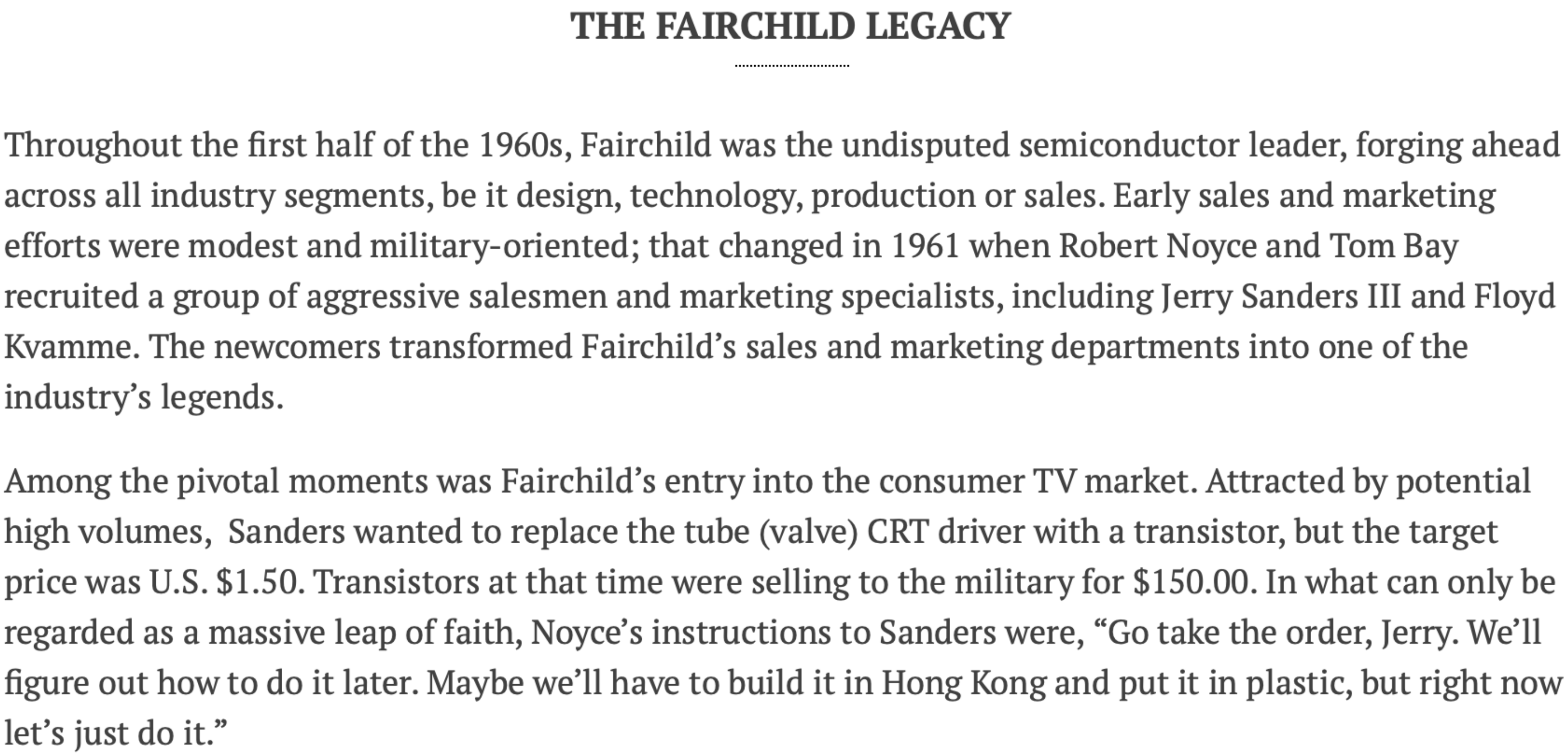
MELTING POT FOR THE FAIRCHILDREN

Sheldon Roberts, Eugene Kleiner, and Jean Hoerni's collective decision to leave and compete against Fairchild, just over three years after the company was founded, was the first of what would be many subsequent defections and spinouts, eventually known as "Fairchildren," directly or indirectly creating dozens of corporations, including Intel and AMD. In doing so, Fairchild sowed the seeds of innovation across multiple companies in the region that would eventually become known as Silicon Valley.

While it is unclear who came up with the moniker, "Silicon Valley," Don Hoefler, a technology reporter for the industry publication *Electronic News*, is often credited with popularizing the name in a 1971 column about the region's chip industry. Hoefler also promoted the area's innovative qualities, and was one of the first writers to chronicle the Northern Californian technology industry as a community.

Don Hoefler

Local watering holes, restaurants and other hot spots provided venues for Silicon Valley's "work hard, play hard" ethos, where industry folk gathered after work to drink, gossip, brag, trade war stories, talk shop, exchange ideas, change jobs and develop new contacts. Key venues included the Wagon Wheel, Lion & Compass, and Ricky's, along with the Peppermill and the Sunnyvale Hilton.

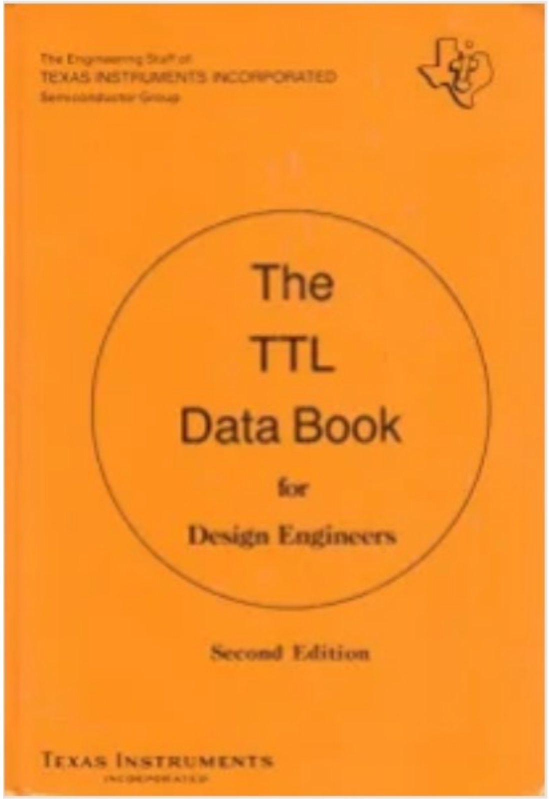


THE FAIRCHILD LEGACY

Throughout the first half of the 1960s, Fairchild was the undisputed semiconductor leader, forging ahead across all industry segments, be it design, technology, production or sales. Early sales and marketing efforts were modest and military-oriented; that changed in 1961 when Robert Noyce and Tom Bay recruited a group of aggressive salesmen and marketing specialists, including Jerry Sanders III and Floyd Kvamme. The newcomers transformed Fairchild's sales and marketing departments into one of the industry's legends.

Among the pivotal moments was Fairchild's entry into the consumer TV market. Attracted by potential high volumes, Sanders wanted to replace the tube (valve) CRT driver with a transistor, but the target price was U.S. $1.50. Transistors at that time were selling to the military for $150.00. In what can only be regarded as a massive leap of faith, Noyce's instructions to Sanders were, "Go take the order, Jerry. We'll figure out how to do it later. Maybe we'll have to build it in Hong Kong and put it in plastic, but right now let's just do it."

TI TTL



The Engineering Staff of
TEXAS INSTRUMENTS INCORPORATED
Semiconductor Group

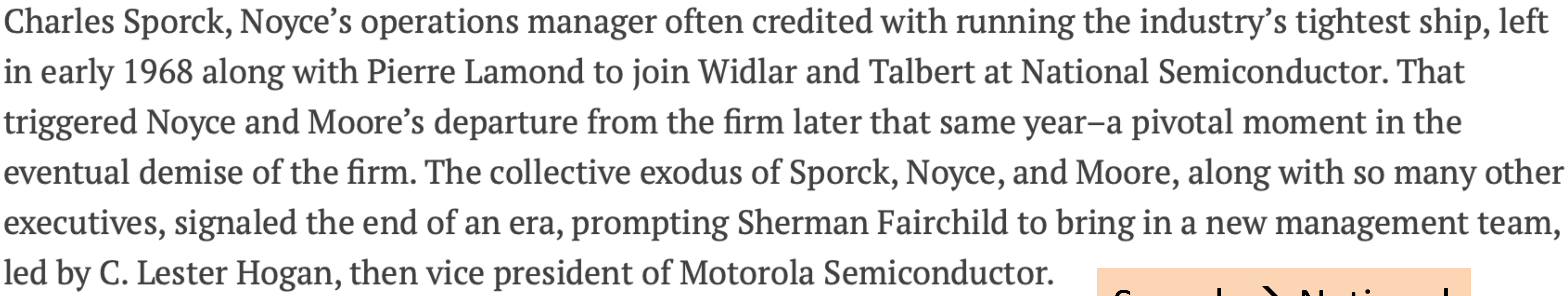
The
TTL
Data Book
for
Design Engineers

Second Edition

Texas Instruments
Incorporated

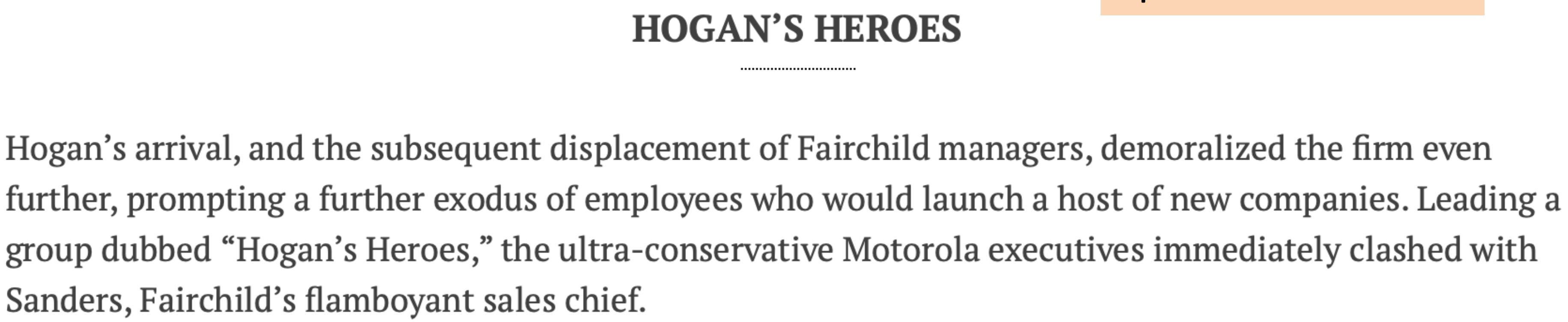
The TTL Data Book for Design Engineers.

By always ensuring any bill of materials included at least one TTL part that was only available from it, Texas Instruments was able to stay one step ahead of the competition and own the T'TL market for the best part of 30 years, until standard logic eventually fell victim to the 1980s application-specific IC revolution.



Charles Sporck, Noyce's operations manager often credited with running the industry's tightest ship, left in early 1968 along with Pierre Lamond to join Widlar and Talbert at National Semiconductor. That triggered Noyce and Moore's departure from the firm later that same year–a pivotal moment in the eventual demise of the firm. The collective exodus of Sporck, Noyce, and Moore, along with so many other executives, signaled the end of an era, prompting Sherman Fairchild to bring in a new management team, led by C. Lester Hogan, then vice president of Motorola Semiconductor.

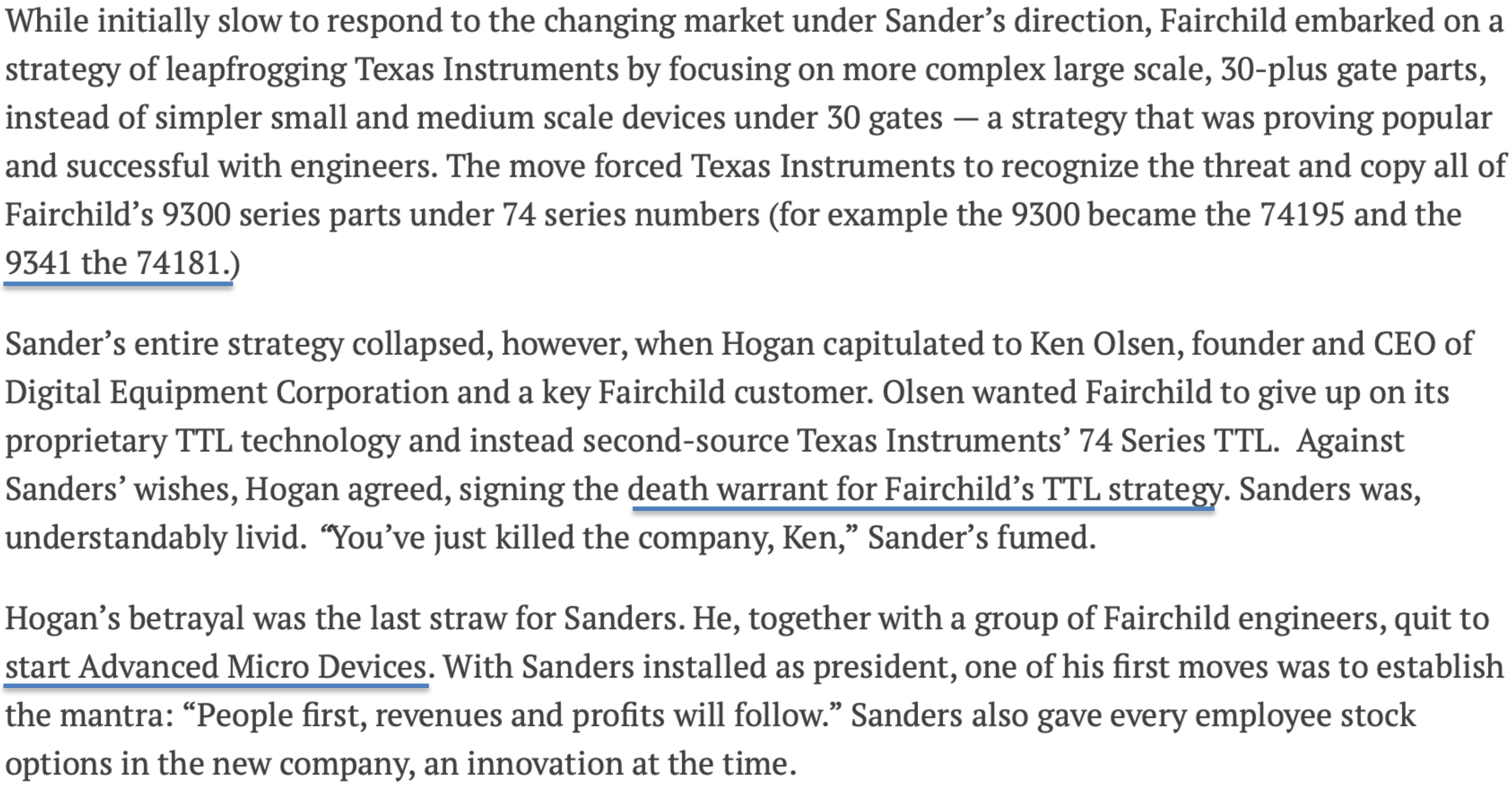
Sporck → National



HOGAN'S HEROES

Hogan's arrival, and the subsequent displacement of Fairchild managers, demoralized the firm even further, prompting a further exodus of employees who would launch a host of new companies. Leading a group dubbed "Hogan's Heroes," the ultra-conservative Motorola executives immediately clashed with Sanders, Fairchild's flamboyant sales chief.

Hogan/Wilf/Sanders



While initially slow to respond to the changing market under Sander's direction, Fairchild embarked on a strategy of leapfrogging Texas Instruments by focusing on more complex large scale, 30-plus gate parts, instead of simpler small and medium scale devices under 30 gates — a strategy that was proving popular and successful with engineers. The move forced Texas Instruments to recognize the threat and copy all of Fairchild's 9300 series parts under 74 series numbers (for example the 9300 became the 74195 and the 9341 the 74181.)

Sander's entire strategy collapsed, however, when Hogan capitulated to Ken Olsen, founder and CEO of Digital Equipment Corporation and a key Fairchild customer. Olsen wanted Fairchild to give up on its proprietary TTL technology and instead second-source Texas Instruments' 74 Series TTL. Against Sanders' wishes, Hogan agreed, signing the death warrant for Fairchild's TTL strategy. Sanders was, understandably livid. "You've just killed the company, Ken," Sander's fumed.

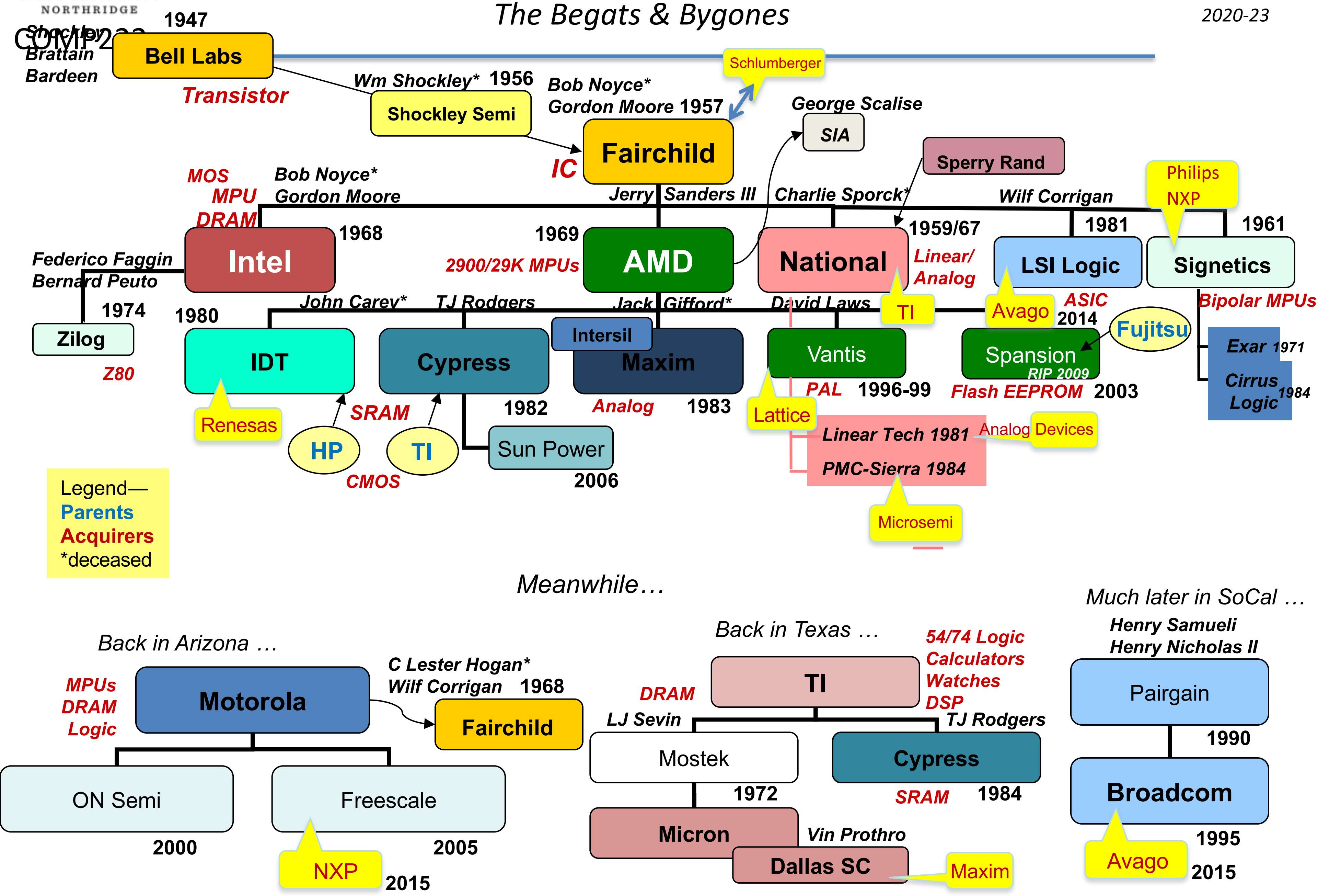
Hogan's betrayal was the last straw for Sanders. He, together with a group of Fairchild engineers, quit to start Advanced Micro Devices. With Sanders installed as president, one of his first moves was to establish the mantra: "People first, revenues and profits will follow." Sanders also gave every employee stock options in the new company, an innovation at the time.

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP2

DR JEFF SOFTWARE
INDIE APP DEVELOPER
© Jeff Drobman
2020-23

Genesis of Silicon Valley

The Begats & Bygones



1947
Shockley
Brattain
Bardeen
Bell Labs
Transistor
Wm Shockley* 1956
Shockley Semi
Bob Noyce*
Gordon Moore 1957
Schlumberger
Fairchild
IC
George Scalise
SIA
Sperry Rand
MOS
MPU
DRAM
Bob Noyce*
Gordon Moore
Jerry Sanders III
Charlie Sporck*
Wilf Corrigan
Philips
NXP
Intel
1968
1969
AMD
2900/29K MPUs
National
1959/67
Linear/
Analog
1981
LSI Logic
1961
Signetics
Federico Faggin
Bernard Peuto
1974
Zilog
Z80
1980
John Carey*
TJ Rodgers
Jack Gifford*
David Laws
TI
Avago
ASIC
2014
Fujitsu
Bipolar MPUs
IDT
Cypress
Intersil
Maxim
Vantis
Spansion
RIP 2009
Exar 1971
Cirrus Logic 1984
Renesas
SRAM
HP
TI
CMOS
1982
Sun Power
2006
Analog
1983
Lattice
PAL
1996-99
Flash EEPROM
2003
Linear Tech 1981
Analog Devices
PMC-Sierra 1984
Microsemi

Legend—
Parents
Acquirers
*deceased

Meanwhile…

Back in Arizona …
MPUs
DRAM
Logic
Motorola
C Lester Hogan*
Wilf Corrigan 1968
Fairchild
ON Semi
2000
Freescale
2005
NXP
2015

Back in Texas …
DRAM
TI
54/74 Logic
Calculators
Watches
DSP
LJ Sevin
TJ Rodgers
Mostek
Cypress
1972
SRAM
1984
Micron
Vin Prothro
Dallas SC
Maxim

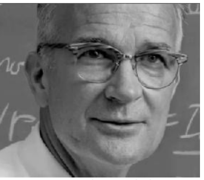
Much later in SoCal …
Henry Samueli
Henry Nicholas II
Pairgain
1990
Broadcom
1995
Avago
2015

Founders HoF

DR JEFF SOFTWARE
INDIE APP DEVELOPER
© Jeff Drobman
2020-23

COMP 222

Bell Labs

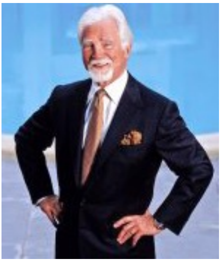


Wm Shockley

Fairchild founders (8)

Wilf Corrigan

Fairchild Chairman/CEO, LSI Logic founder



Jerry Sanders

CEO, AMD
1969–2002

Jack Gifford

AMD co-founder

In 1983, Gifford co-founded [Maxim Integrated Products]

From left: W. Jerry Sanders III, President and Chairman of the Board. D. John Carey, Managing Director of Complex Digital Operations. Sven E. Simonsen, Director of Engineering, Complex Digital Operations. Frank T. Botte, Director of Development, Analog Operations. James N. Giles, Director of Engineering, Analog Operations. Edwin J. Turney, Director of Sales and Administration. Jack F. Gifford, Director of Marketing and Business Development. R. Lawrence Stenger, Managing Director, Analog Operations.

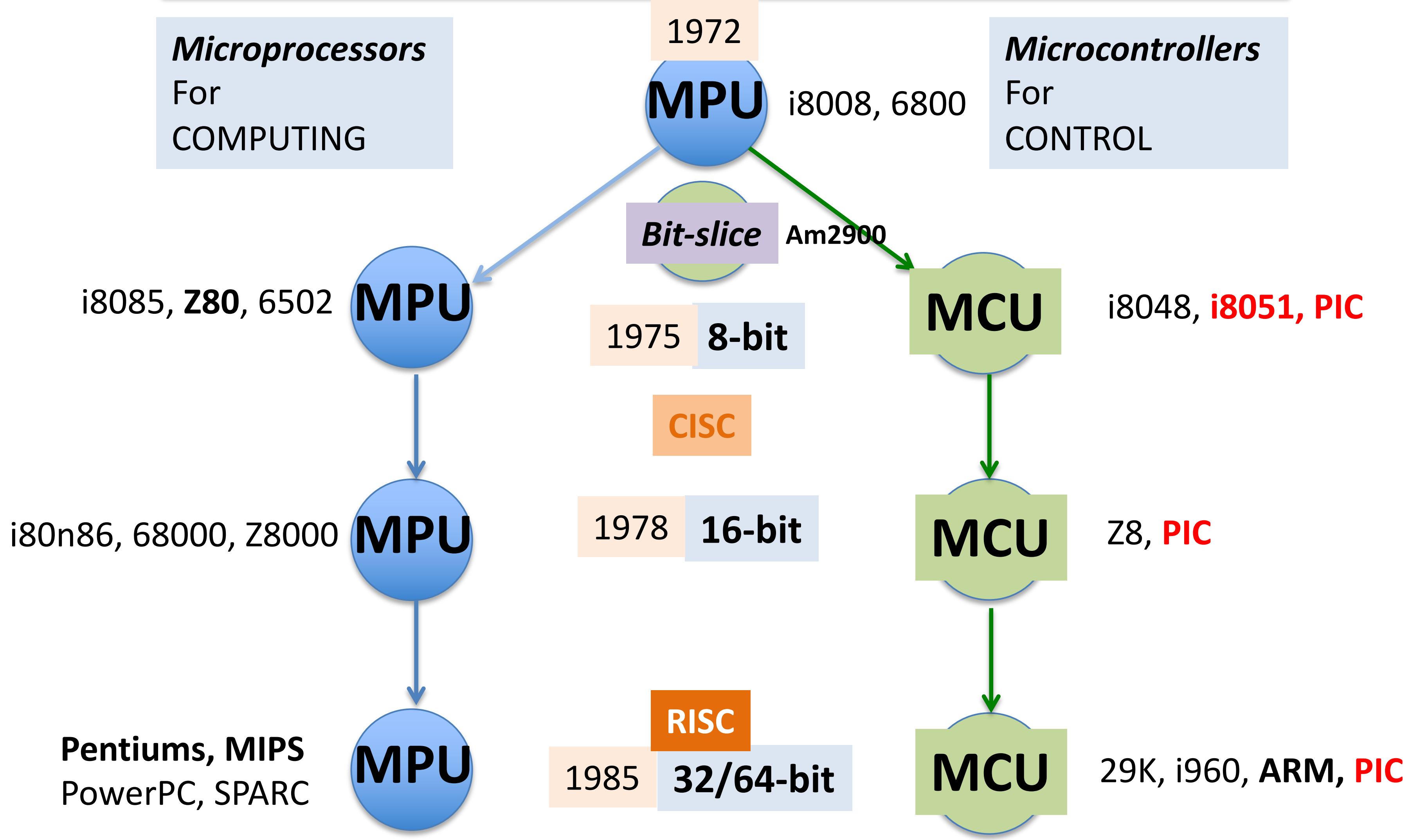
Bob Noyce

Gordon Moore

T. J. Rodgers

Cypress Semi founder

MPU/MCU Generations



Microprocessors
For
COMPUTING

Microcontrollers
For
CONTROL

1972 MPU i8008, 6800

Bit-slice **Am2900**

i8085, **Z80**, 6502 **MPU**

1975 **8-bit**

MCU i8048, **i8051, PIC**

CISC

i80n86, 68000, Z8000 **MPU**

1978 **16-bit**

MCU Z8, **PIC**

Pentiums, MIPS
PowerPC, SPARC
MPU

RISC
1985 **32/64-bit**

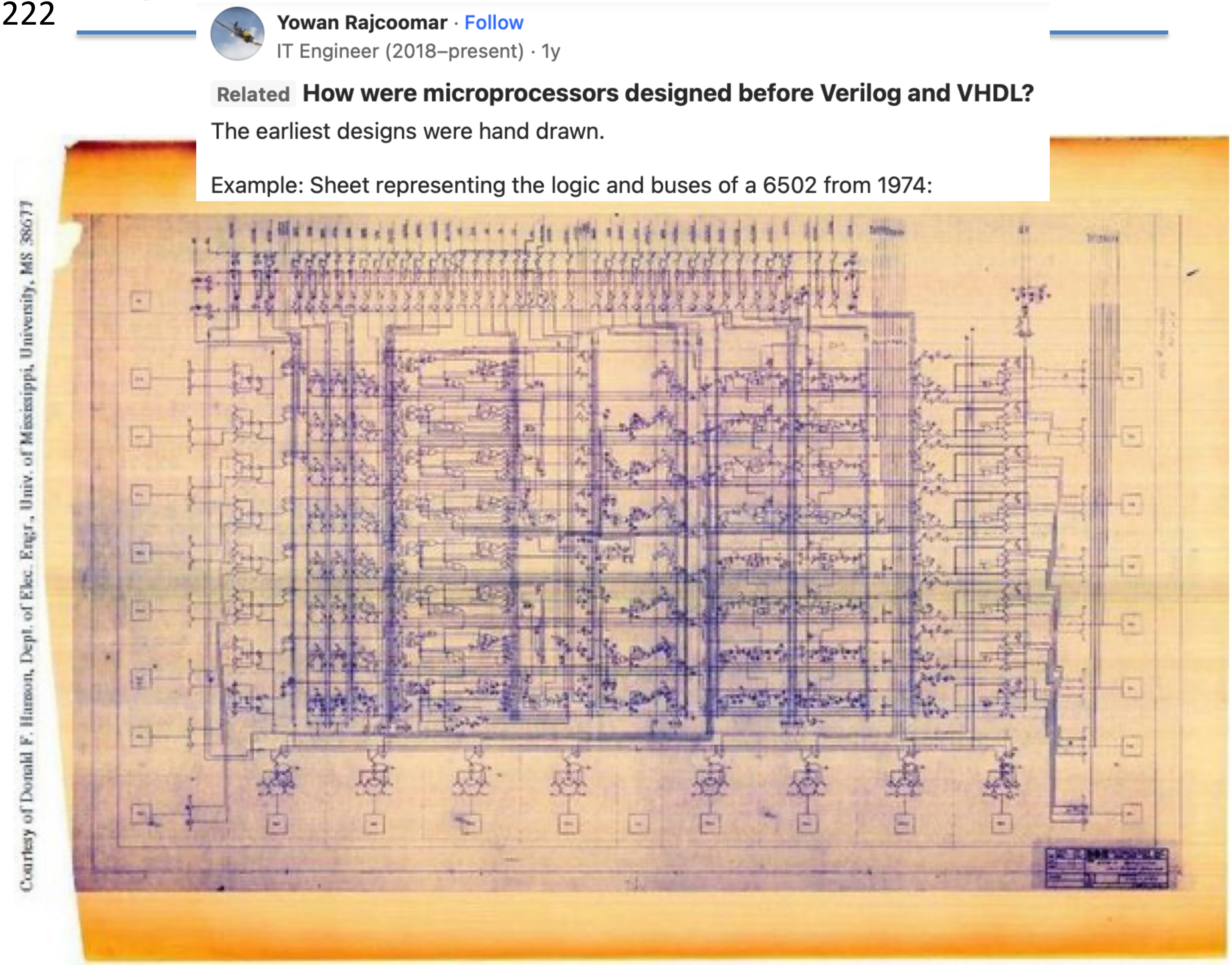
MCU 29K, i960, **ARM**, **PIC**

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP222

Quora

6502 in 1974

DR JEFF SOFTWARE
INDIE APP DEVELOPER



Yowan Rajcoomar · Follow
IT Engineer (2018–present) · 1y

Related **How were microprocessors designed before Verilog and VHDL?**

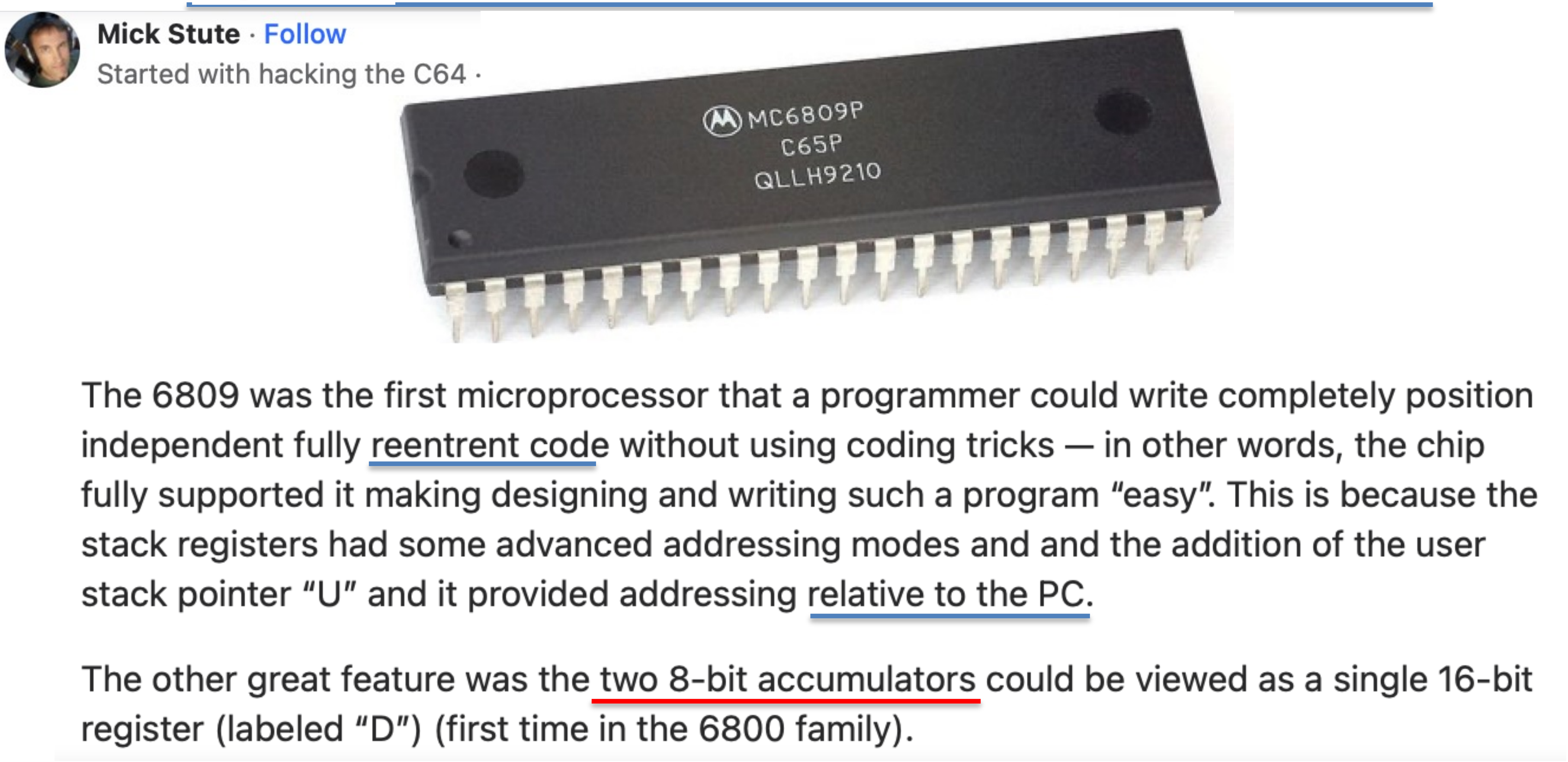
The earliest designs were hand drawn.

Example: Sheet representing the logic and buses of a 6502 from 1974:

Courtesy of Donald F. Hanson, Dept. of Elec. Engr., Univ. of Mississippi, University, MS 38677

M6809

Quora



Mick Stute · Follow
Started with hacking the C64 ·

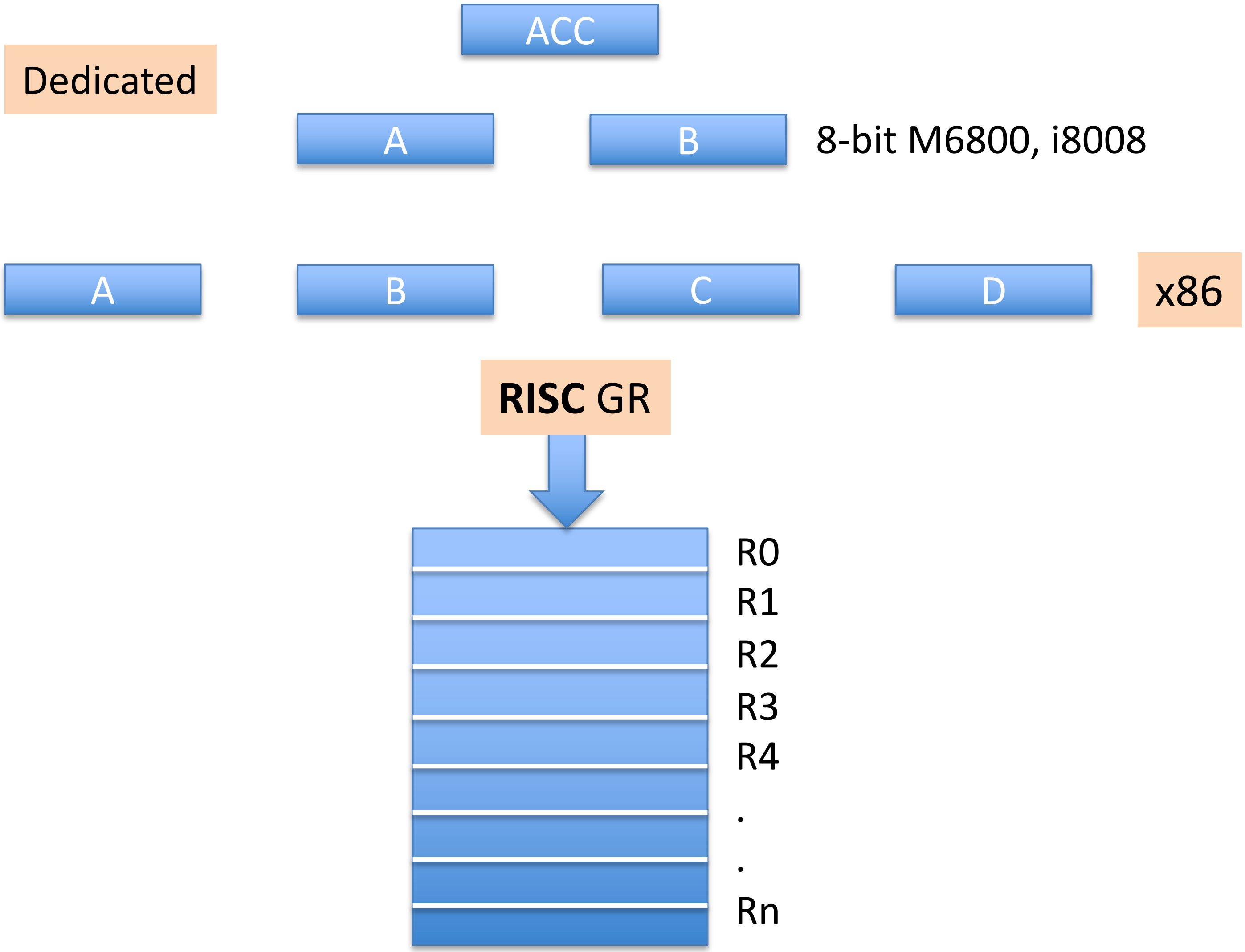
The 6809 was the first microprocessor that a programmer could write completely position independent fully reentrent code without using coding tricks — in other words, the chip fully supported it making designing and writing such a program “easy”. This is because the stack registers had some advanced addressing modes and and the addition of the user stack pointer “U” and it provided addressing relative to the PC.

The other great feature was the two 8-bit accumulators could be viewed as a single 16-bit register (labeled “D”) (first time in the 6800 family).

A

B

Accumulator Evolution



ACC

Dedicated

A

B

8-bit M6800, i8008

A

B

C

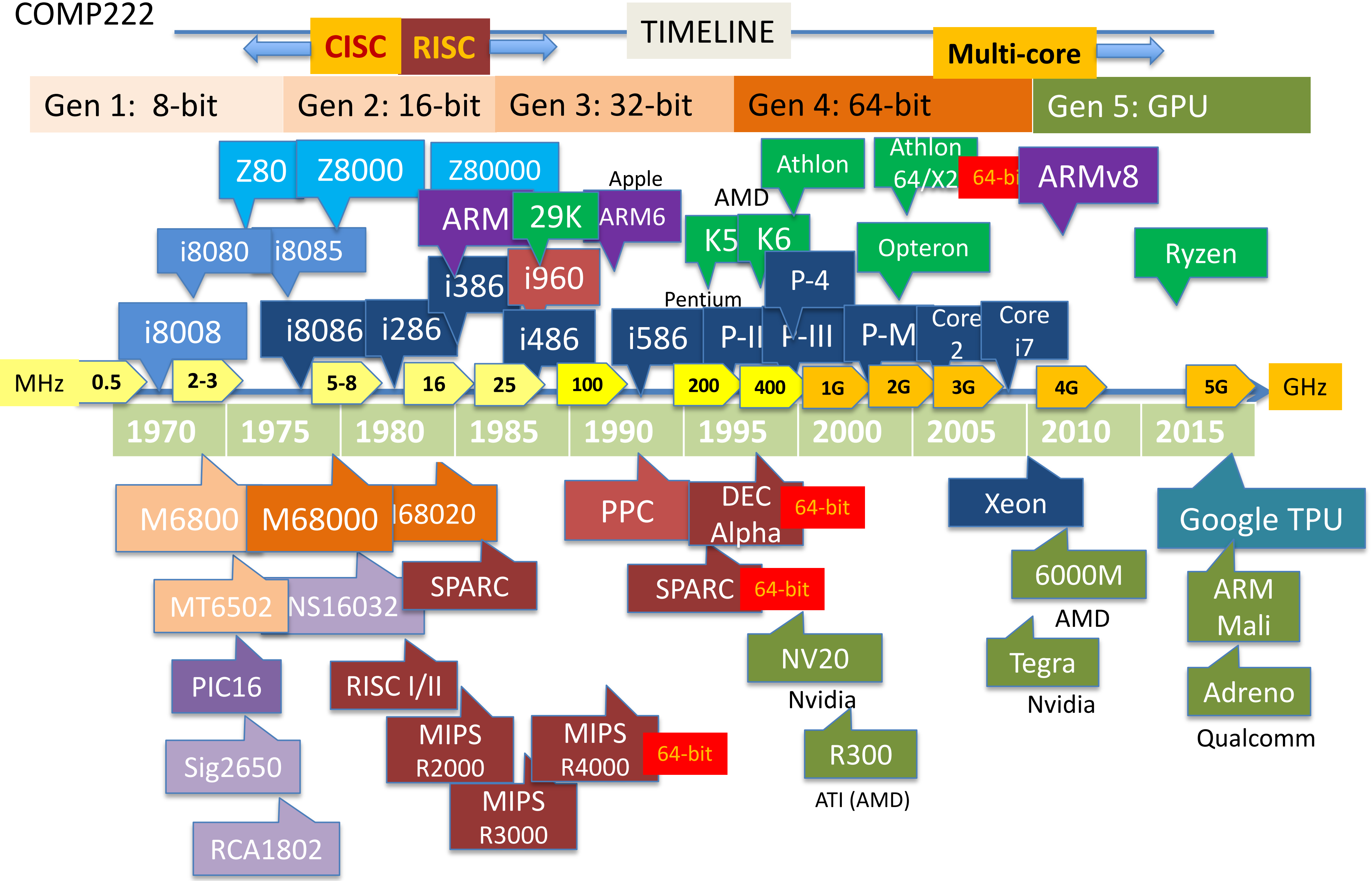
D

x86

RISC GR

R0
R1
R2
R3
R4
.
.
Rn

MPU Generations



TIMELINE

CISC RISC

Multi-core

Gen 1: 8-bit

Gen 2: 16-bit

Gen 3: 32-bit

Gen 4: 64-bit

Gen 5: GPU

Z80 Z8000 Z80000

i8080 i8085

i8008

i8086 i286

ARM 29K

Apple ARM6

i386 i960

i486

Pentium

i586 P-II P-III

AMD

K5 K6

P-4

Athlon

Athlon 64/X2 64-bi

Opteron

P-M Core 2 Core i7

ARMv8

Ryzen

MHz 0.5 2-3 5-8 16 25 100 200 400 1G 2G 3G 4G 5G GHz

1970 1975 1980 1985 1990 1995 2000 2005 2010 2015

M6800 M68000 68020

MT6502 NS16032 SPARC

PIC16

Sig2650

RCA1802

RISC I/II

MIPS R2000

MIPS R3000

MIPS R4000 64-bit

PPC

DEC Alpha 64-bit

SPARC 64-bit

NV20 Nvidia

R300 ATI (AMD)

Xeon

6000M AMD

Tegra Nvidia

Google TPU

ARM Mali

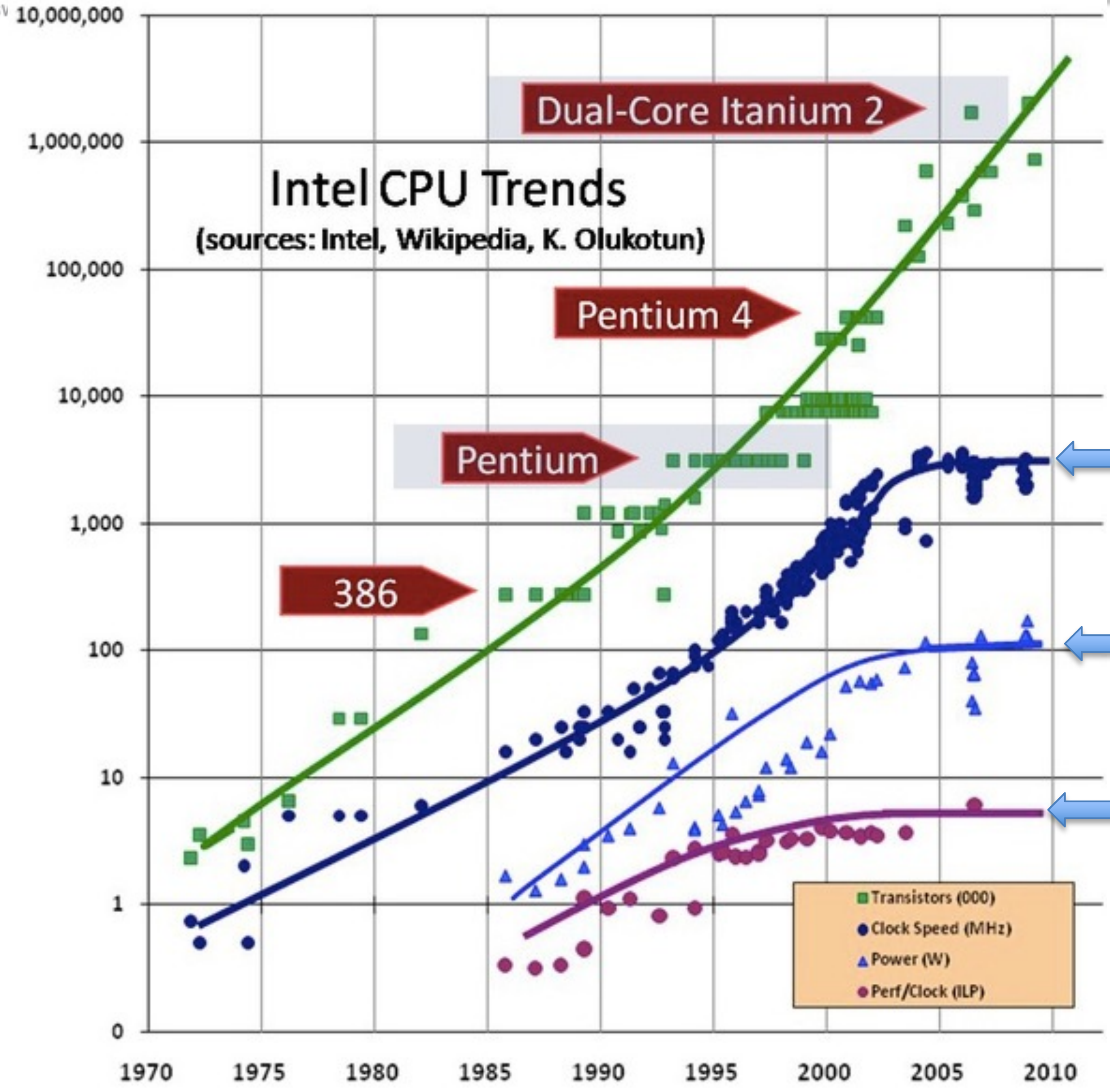
Adreno Qualcomm

Quora

CPU Trends

Steve Baker, Blogger at LetsRunWithIt.com (2013-present)

Answ



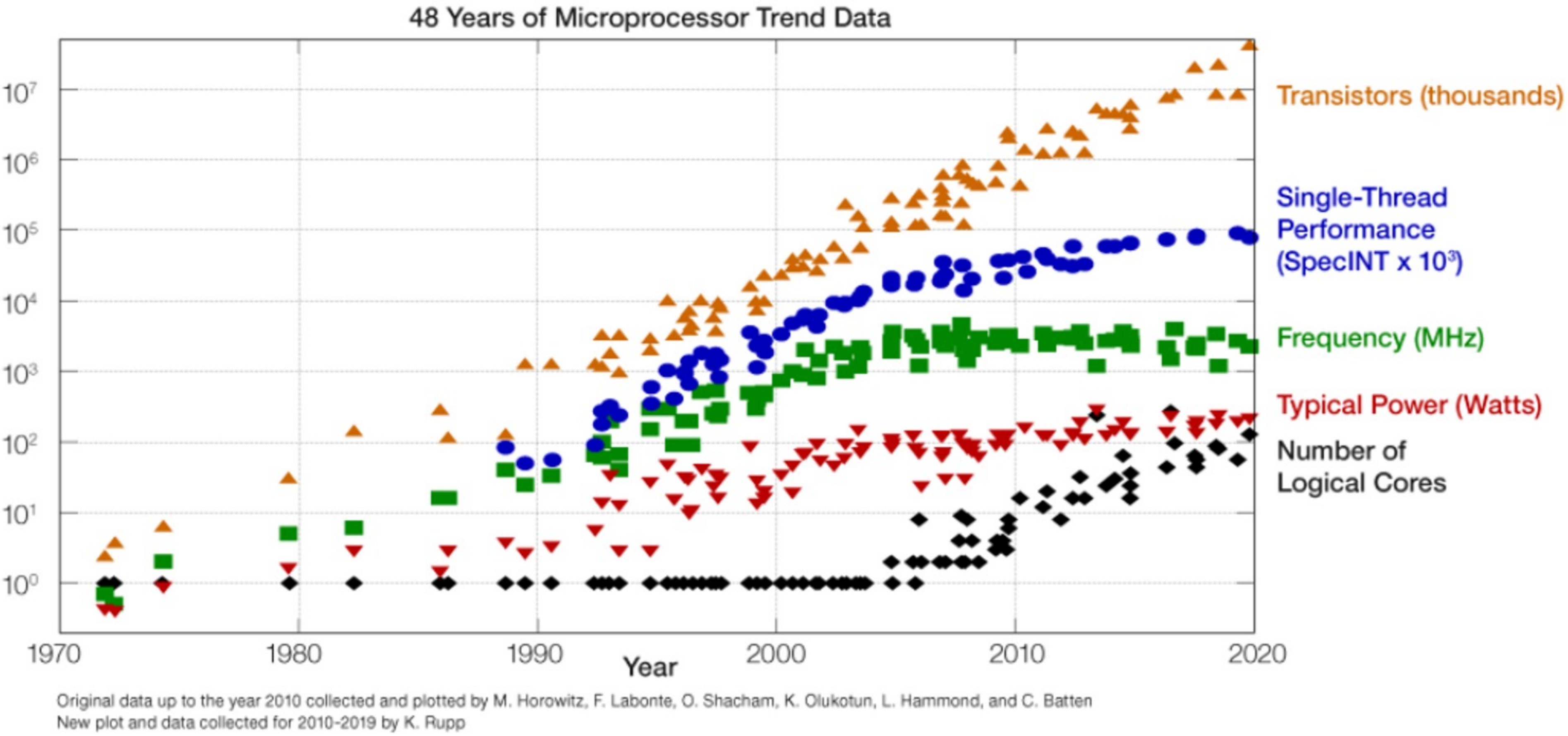
5 GHz

100 W

CPI/IPC

Micro-arch

CPU Trends



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2019 by K. Rupp

Timeline of CPU Speed

Multi-Core

MT

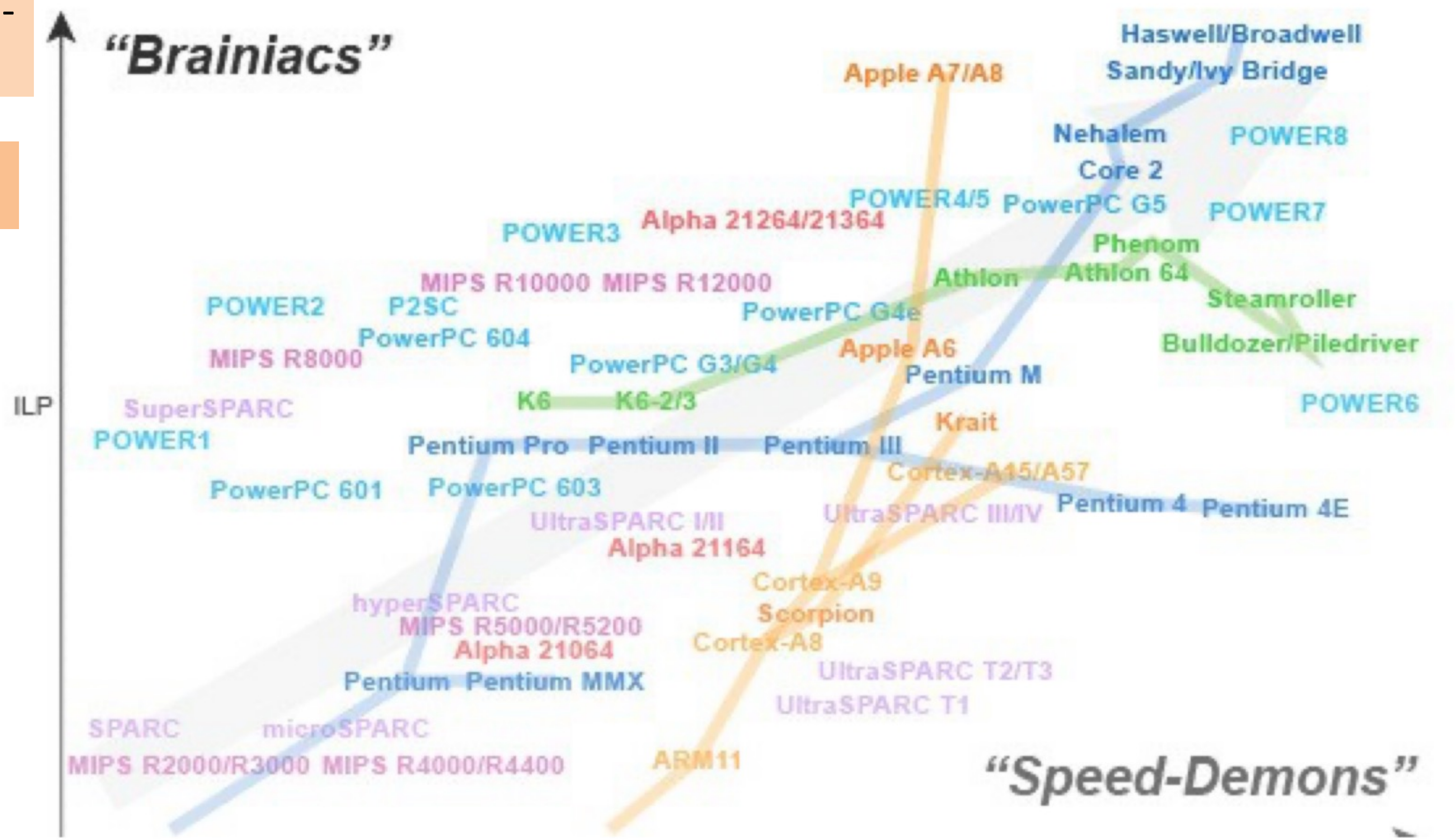
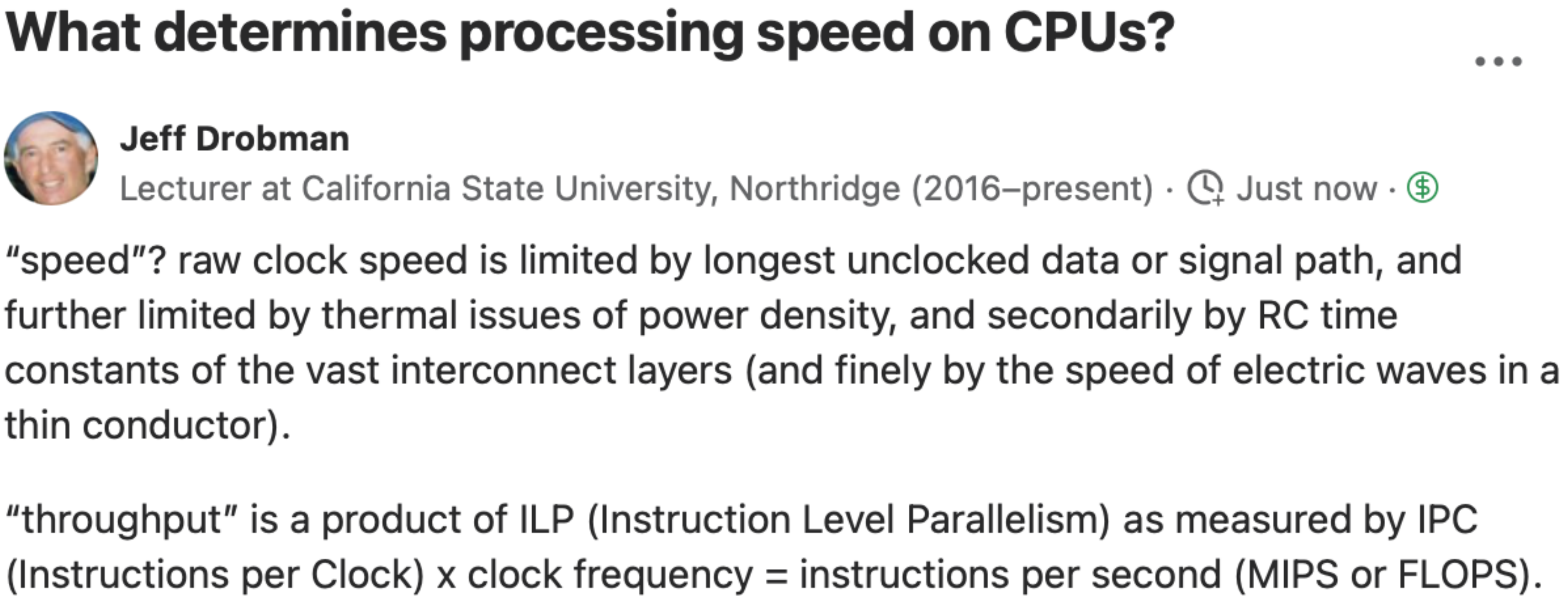


image: CPU Architecture War : Clock Speed v.s. Instruction-Level Parallelism [https://www.linkedin.com/pulse/cpu-architecture-war-clock-speed-vs-instruction-level-hanindhito]

CPU Speed



What determines processing speed on CPUs?

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

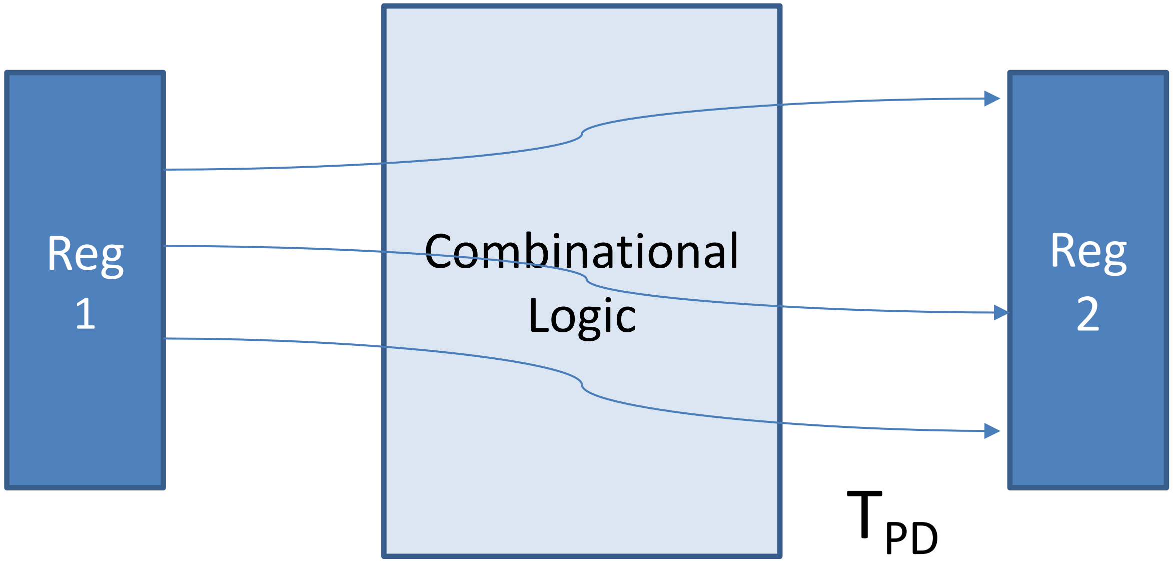
“speed”? raw clock speed is limited by longest unclocked data or signal path, and further limited by thermal issues of power density, and secondarily by RC time constants of the vast interconnect layers (and finely by the speed of electric waves in a thin conductor).

“throughput” is a product of ILP (Instruction Level Parallelism) as measured by IPC (Instructions per Clock) x clock frequency = instructions per second (MIPS or FLOPS).

MIPS = **IPC*** x F

*Total **IPC** = IPC per core x N (cores)

Cycle Time Limit: Slowest Path

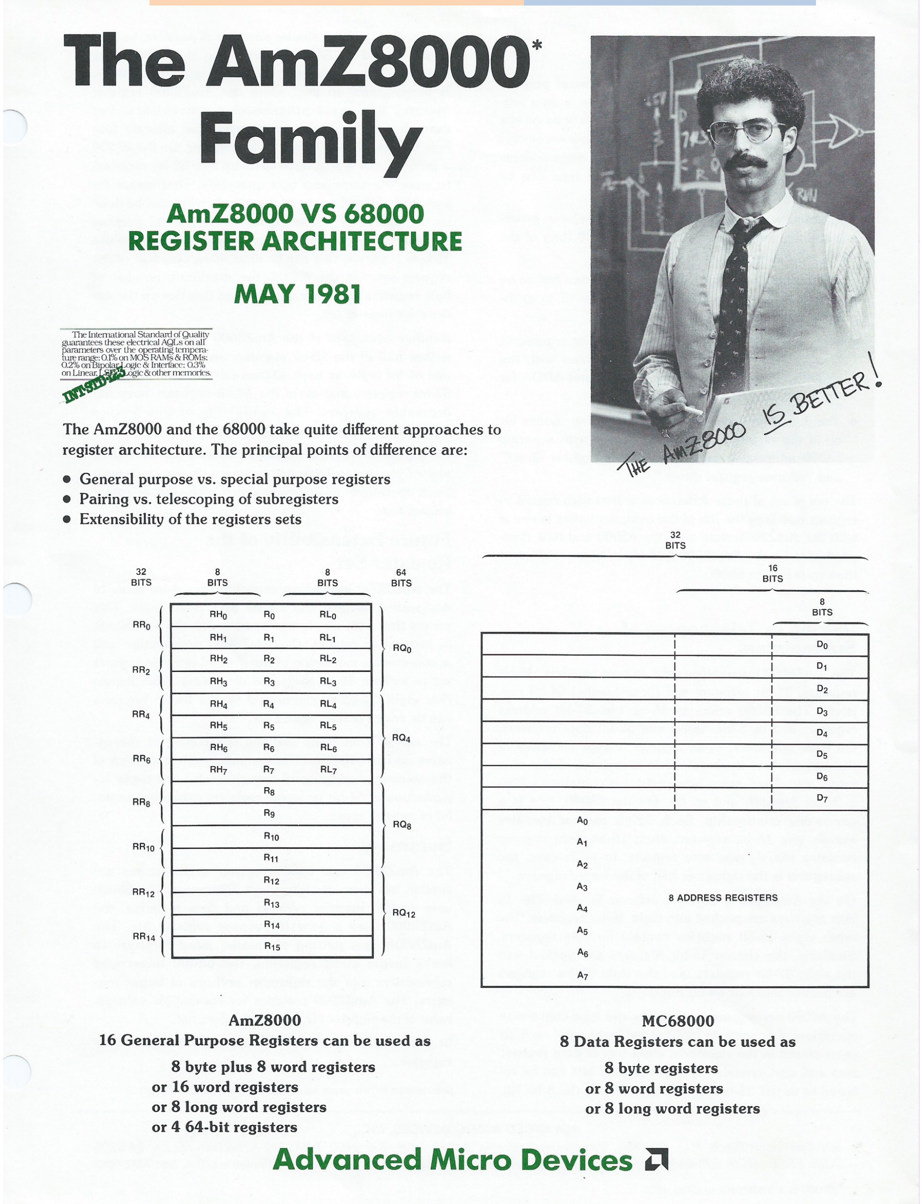


$$T_{CYC} = T_{CQ1} + T_{PD} + T_{SU2}$$

CPU ISA's

Z8000 vs. M6800 | 16-bit MPU's

- ❖x86
  - ❑ i8088
  - ❑ Pentium
    - ▪ Intel P, M
    - ▪ AMD K5-8
- ❖MIPS
  - ❑ R3000/4000
  - ❑ MIPS32/64
- ❖ARM
  - ❑ Cortex (A, M)
  - ❑ ARMv7/8
- ❖RISC-V



The AmZ8000* Family

AmZ8000 VS 68000
REGISTER ARCHITECTURE

MAY 1981

The International Standard of Quality guarantees these electrical AQLs on all parameters over the operating temperature range: 0.1% on MOS RAMs & ROMs: 0.2% on Bipolar Logic & Interface: 0.3% on Linear, LSI Logic & other memories.

INT·STD·123

THE AmZ8000 IS BETTER!

The AmZ8000 and the 68000 take quite different approaches to register architecture. The principal points of difference are:

- General purpose vs. special purpose registers
- Pairing vs. telescoping of subregisters
- Extensibility of the registers sets

32 BITS	8 BITS		8 BITS	64 BITS
RR_0	RH_0	R_0	RL_0	RQ_0
	RH_1	R_1	RL_1	
RR_2	RH_2	R_2	RL_2	
	RH_3	R_3	RL_3	
RR_4	RH_4	R_4	RL_4	RQ_4
	RH_5	R_5	RL_5	
RR_6	RH_6	R_6	RL_6	
	RH_7	R_7	RL_7	
RR_8		R_8		RQ_8
		R_9		
RR_{10}		R_{10}		
		R_{11}		
RR_{12}		R_{12}		RQ_{12}
		R_{13}		
RR_{14}		R_{14}		
		R_{15}		

AmZ8000
16 General Purpose Registers can be used as

8 byte plus 8 word registers
or 16 word registers
or 8 long word registers
or 4 64-bit registers

32 BITS	16 BITS	8 BITS
		D_0
		D_1
		D_2
		D_3
		D_4
		D_5
		D_6
		D_7
A_0	8 ADDRESS REGISTERS	
A_1		
A_2		
A_3		
A_4		
A_5		
A_6		
A_7		

MC68000
8 Data Registers can be used as

8 byte registers
or 8 word registers
or 8 long word registers

Advanced Micro Devices

Section

RISC

vs

CISC

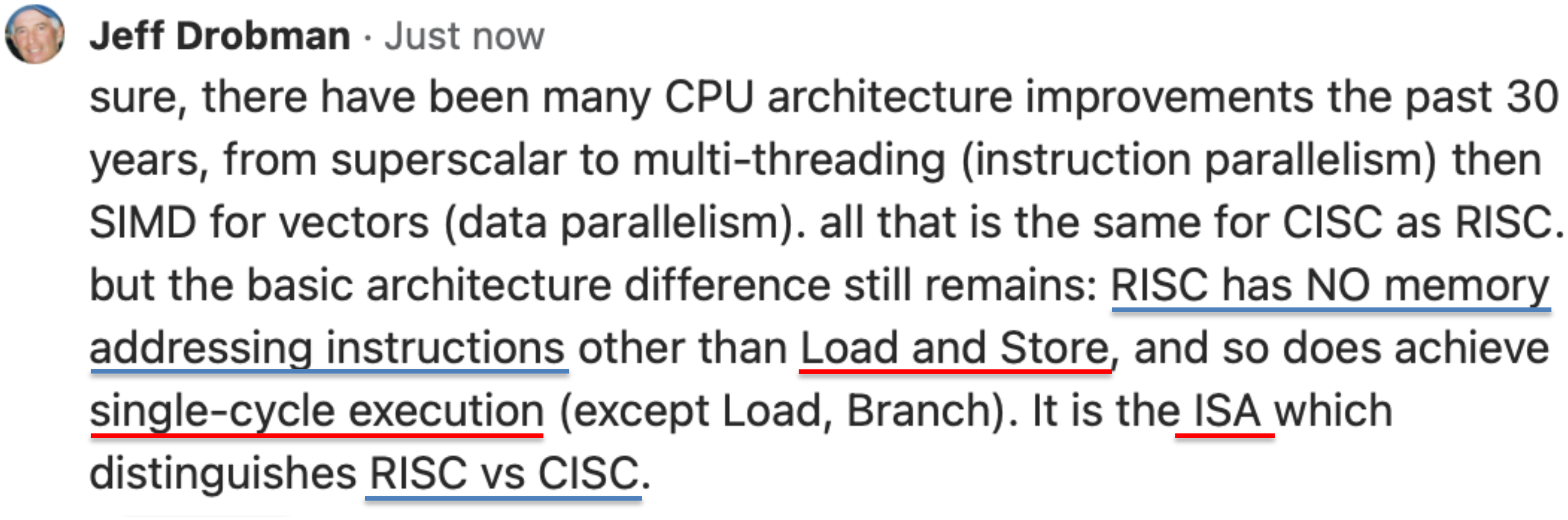
CISC vs RISC:

Complex/Reduced Instruction Set Architecture

- ❖ Microprocessor History
  - ➢ 1971-85: **CISC** (8/16-bit)
    - ✧ Intel i4004 (4-bit)
    - ✧ Intel i8008 (8-bit) → i8080 → i8085, Z80 → i8086 (16-bit) → “x86”
    - ✧ Motorola 6800 (8-bit) → 6502 → 68000 (16-bit)
    - ✧ IBM PC used i8088 (8/16-bit) in 1981 → i80n86 (“x86”) → *Pentiums* (now RISC)
  - ➢ 1985-2000: **RISC** – (32/64-bit)
    - ✧ SPARC* (UC Berkeley→ Sun/Oracle)
    - ✧ MIPS* (Stanford)
    - ✧ PowerPC (Motorola/IBM)
    - ✧ AMD 29K
    - ✧ Intel i960
    - ✧ ARM*

*still exist

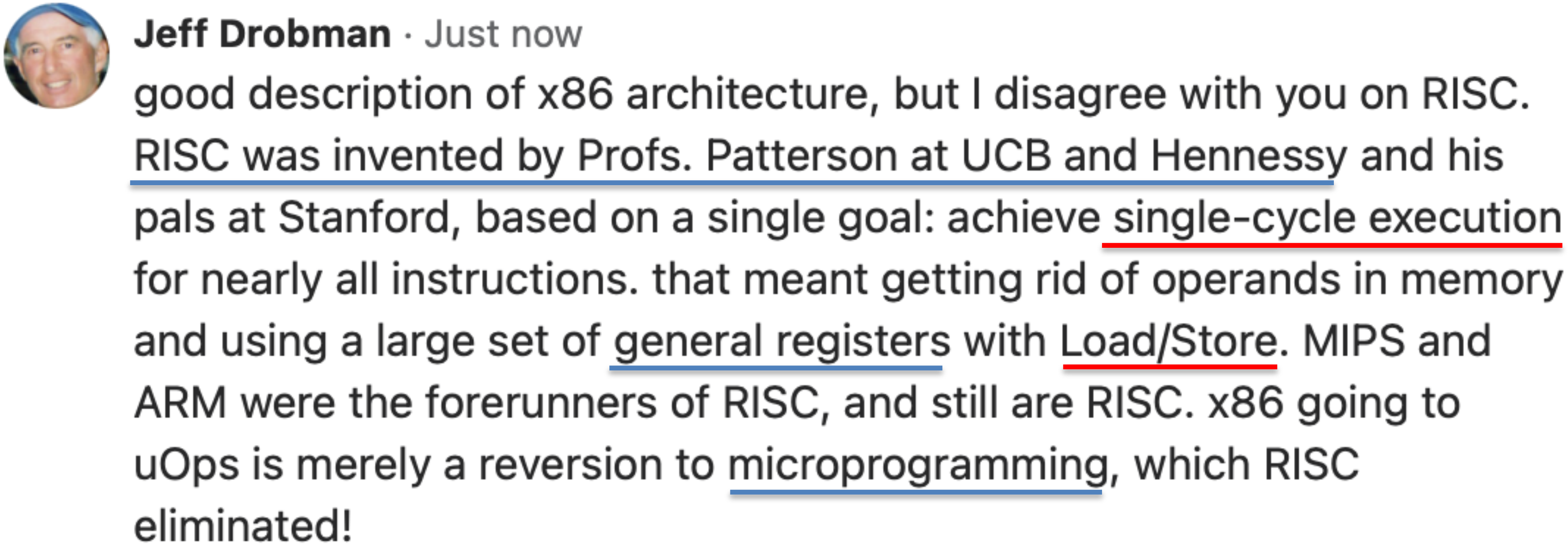
RISC vs CISC



Jeff Drobman · Just now

sure, there have been many CPU architecture improvements the past 30 years, from superscalar to multi-threading (instruction parallelism) then SIMD for vectors (data parallelism). all that is the same for CISC as RISC. but the basic architecture difference still remains: RISC has NO memory addressing instructions other than Load and Store, and so does achieve single-cycle execution (except Load, Branch). It is the ISA which distinguishes RISC vs CISC.

RISC vs CISC



Jeff Drobman · Just now

good description of x86 architecture, but I disagree with you on RISC. RISC was invented by Profs. Patterson at UCB and Hennessy and his pals at Stanford, based on a single goal: achieve single-cycle execution for nearly all instructions. that meant getting rid of operands in memory and using a large set of general registers with Load/Store. MIPS and ARM were the forerunners of RISC, and still are RISC. x86 going to uOps is merely a reversion to microprogramming, which RISC eliminated!

x86 Addresses – LEA

Compare to **MIPS** “la”

- **MOV** vs **LEA** vs **Load/Store**

```
MOV AX,addr  ⬄  LEA AX,[addr]
```

Load AX

```
MOV CX,BX  ⬄  LEA CX,[BX+10]
ADD CX,10
```

*Flags unchanged

Compact Add 10

```
LEA EAX,[EAX+4*EAX] ⬄ MULT EAX,5
```

Multiply by 5

Section

Moore's Law

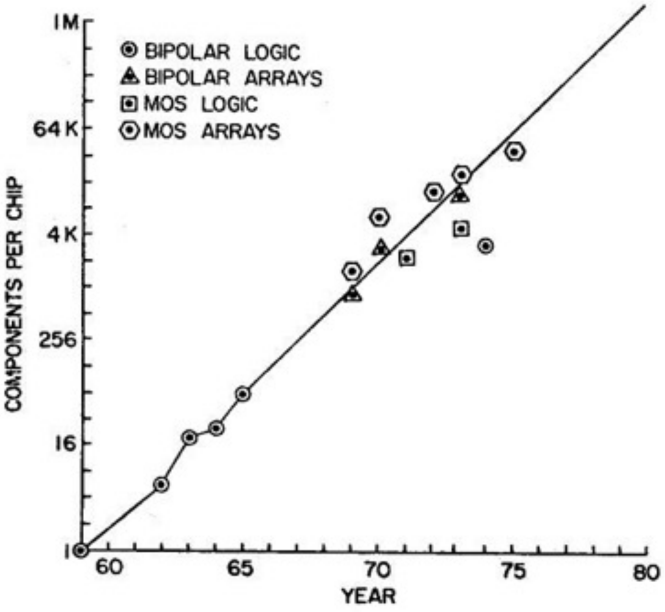
More Moore's Law in separate set: ***Chips & Fabs***

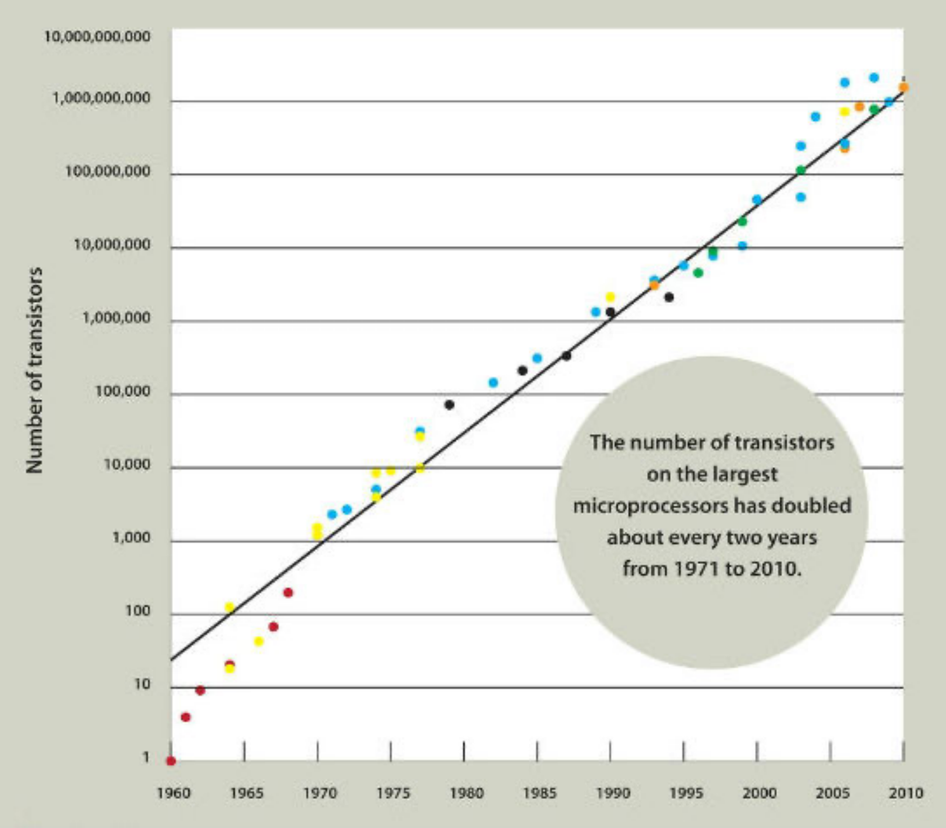
Moore’s Law

DR JEFF SOFTWARE
INDIE APP DEVELOPER

Looking Back

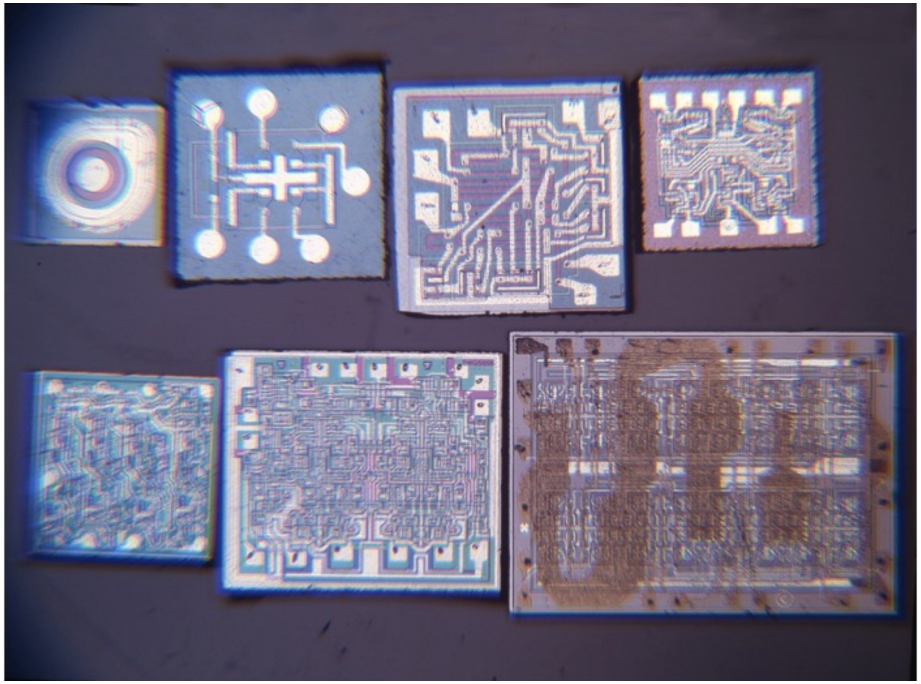
- Original in 1965: # Transistors will double **every year** (12 months)
- Moore revised his prediction in 1975: double **every 2 years** (24 months)
  - → THIS IS MOORE’S LAW
- Intel’s exec David House added CPU complexity would double **every 18 months**
- History shows # Transistors has doubled every –
  - □ **2 years** in *logic*
  - □ **18 months** in DRAM/SRAM





The number of transistors on the largest microprocessors has doubled about every two years from 1971 to 2010.

Origin of Moore's Law



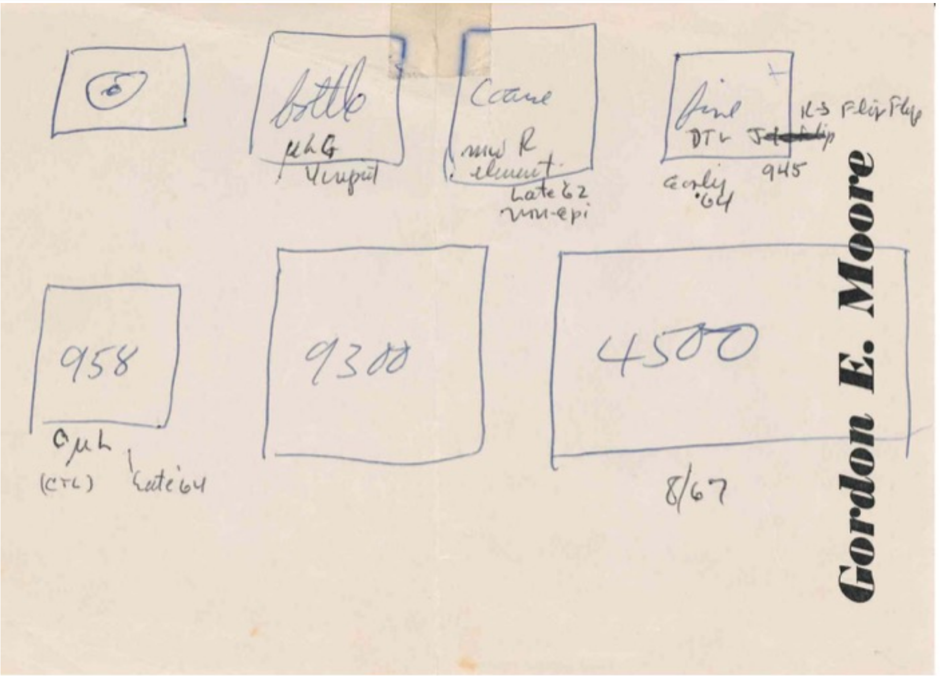

bttle
μLG
4 input
Coarse
mw R
element
Late 62
non-epi
fine
DTL ~~J flip~~
R-S Flip Flop
945
early '64
958
μL
(CTL)
late '64
9300
4500
8/67
Gordon E. Moore

Figure 3. Gordon Moore notes on IC device types. Collection of the Computer History Museum, 102783359.

Year	Device	Function	Transistors	Resistors	Components	LOG_2
1959	2N697	Transistor	1	0	1	0
1962	Type G	RTL 3 - I/P gate	3	4	7	2.8
1963 (late 62)	Type R	RTL D Flip Flop	15	18	33	5.0
1964	945	DTL R-S Flip Flop	13	21	34	5.1
1965 (late 64)	958	RTL Counter	33	25	58	5.9
1966	9300	TTL Shift Register	85	40	125	7.0
1967	4500	DTL 32- Gate Array	200	64	264*	8.0

Figure 4. Table of component count for devices in photograph.

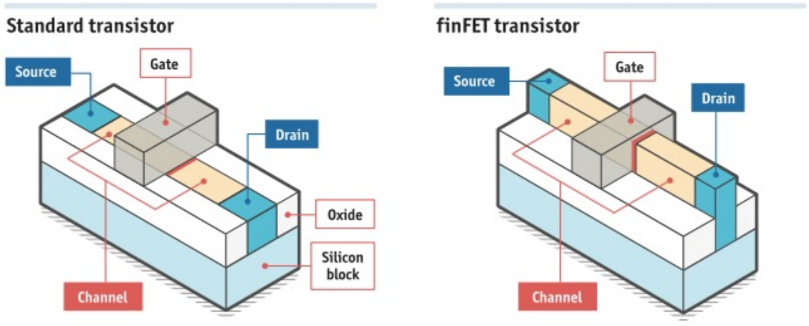
Future of Moore’s Law

Looking Forward

- ❖ Chip design
  - ❑ Transistors
    - ▪ SiGe
    - ▪ FinFET
    - ▪ JNT
  - ❑ Chip stacks (3D hybrid)
    - ▪ Intel/Micron 3D Xpoint
  - ❑ 3D
    - ▪ NAND Flash (EEPROM)
- ❖ Architecture
  - ❑ Specialized hardware (GPU, APU, etc.)
  - ❑ Reconfigurable hardware (FPGA)
- ❖ Thermal/Cooling
  - ❑ Microfluidics (liquid cooling)
- ❖ Something completely diffferent
  - ❑ Molecular computing
  - ❑ Quantum computing

> *“As Moore’s Law slows, we are being forced to make tough choices between Power, Performance and Cost.” (ARM)*

Better by design



A transistor is a switch. Ordinarily, current cannot flow. When a voltage is applied to the **gate**, the **channel** becomes conductive, current flows from the **source** to the **drain**, and the transistor switches on.

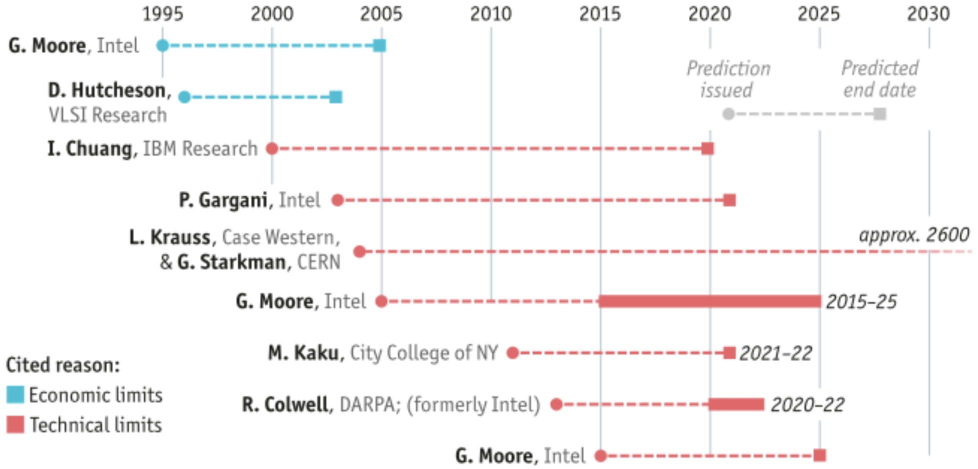
A finFET transistor raises the **channel** above the block of silicon upon which the device sits. That allows the **gate** to wrap around three sides of the **channel**, improving its electrical properties.

Source: *The Economist*

New sorts of transistors can eke out a few more iterations of Moore’s law, but they will get increasingly expensive

Faith no Moore

Selected predictions for the end of Moore’s law



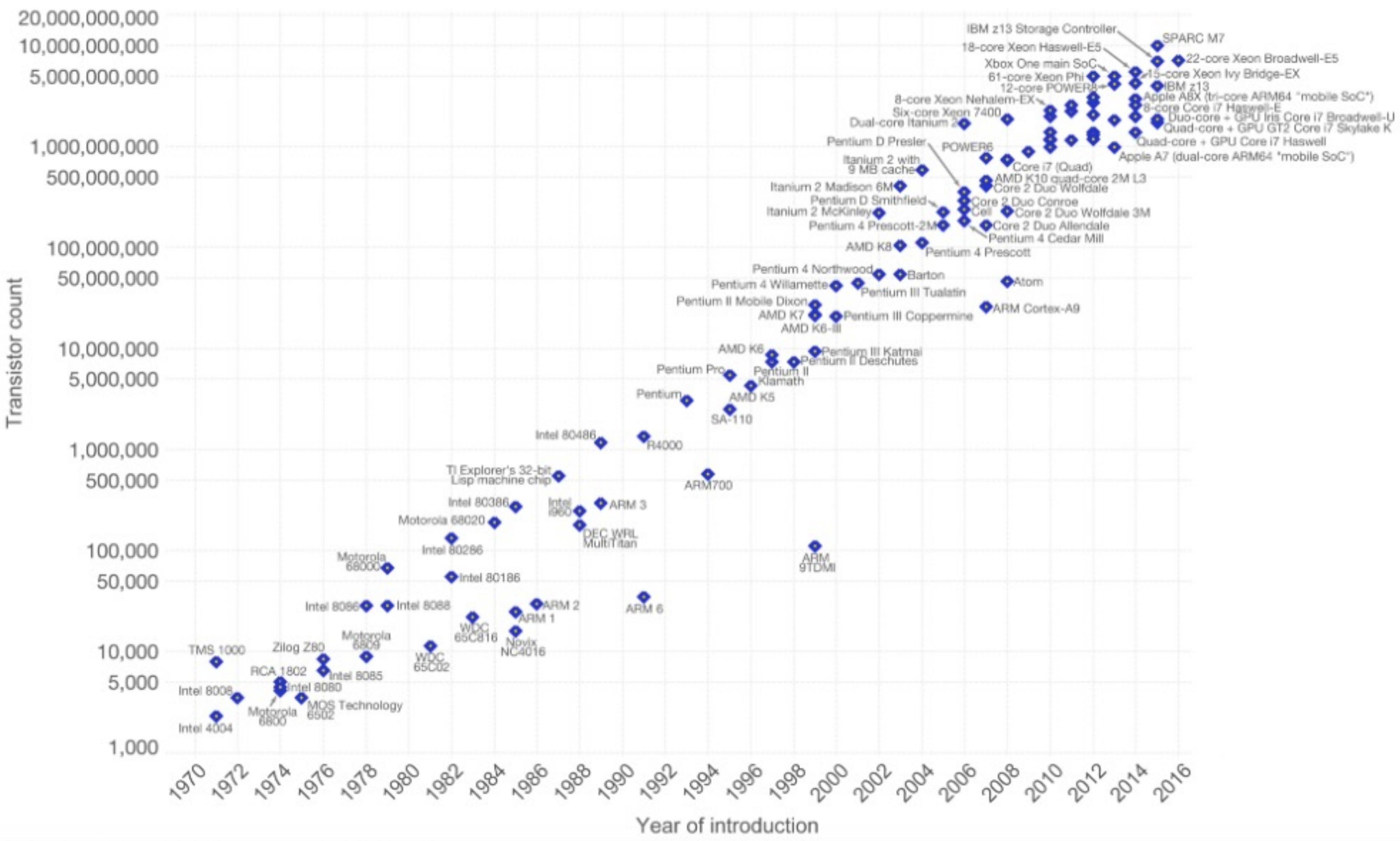
Sources: Intel; press reports; *The Economist*

Microprocessor Timeline

Moore's Law – The number of transistors on integrated circuit chips (1971-2016)

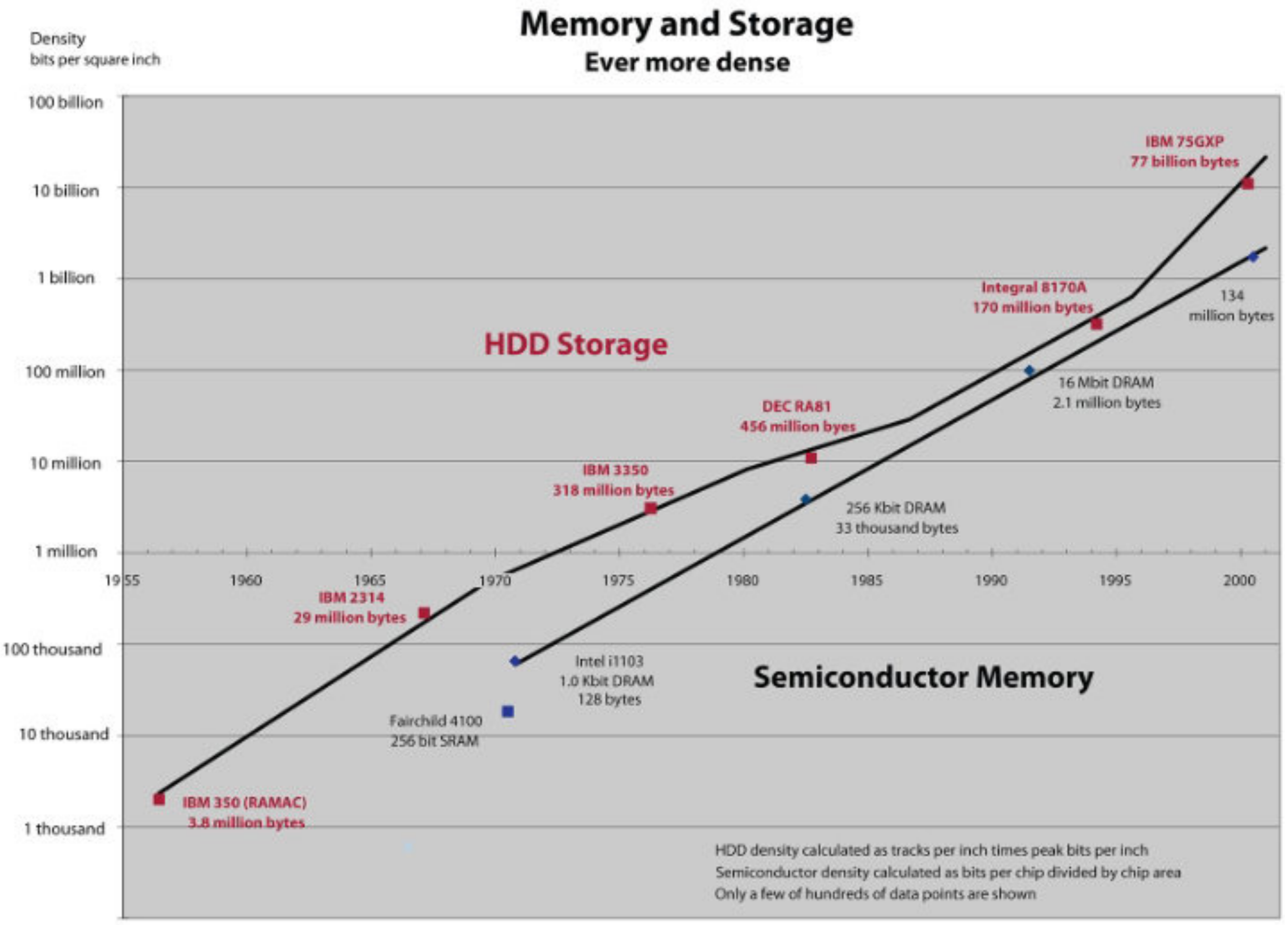
Our World in Data

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important as other aspects of technological progress – such as processing speed or the price of electronic products – are strongly linked to Moore's law.



Data source: Wikipedia (https://en.wikipedia.org/wiki/Transistor_count)
The data visualization is available at OurWorldinData.org. There you find more visualizations and research on this topic.

Memory Timeline

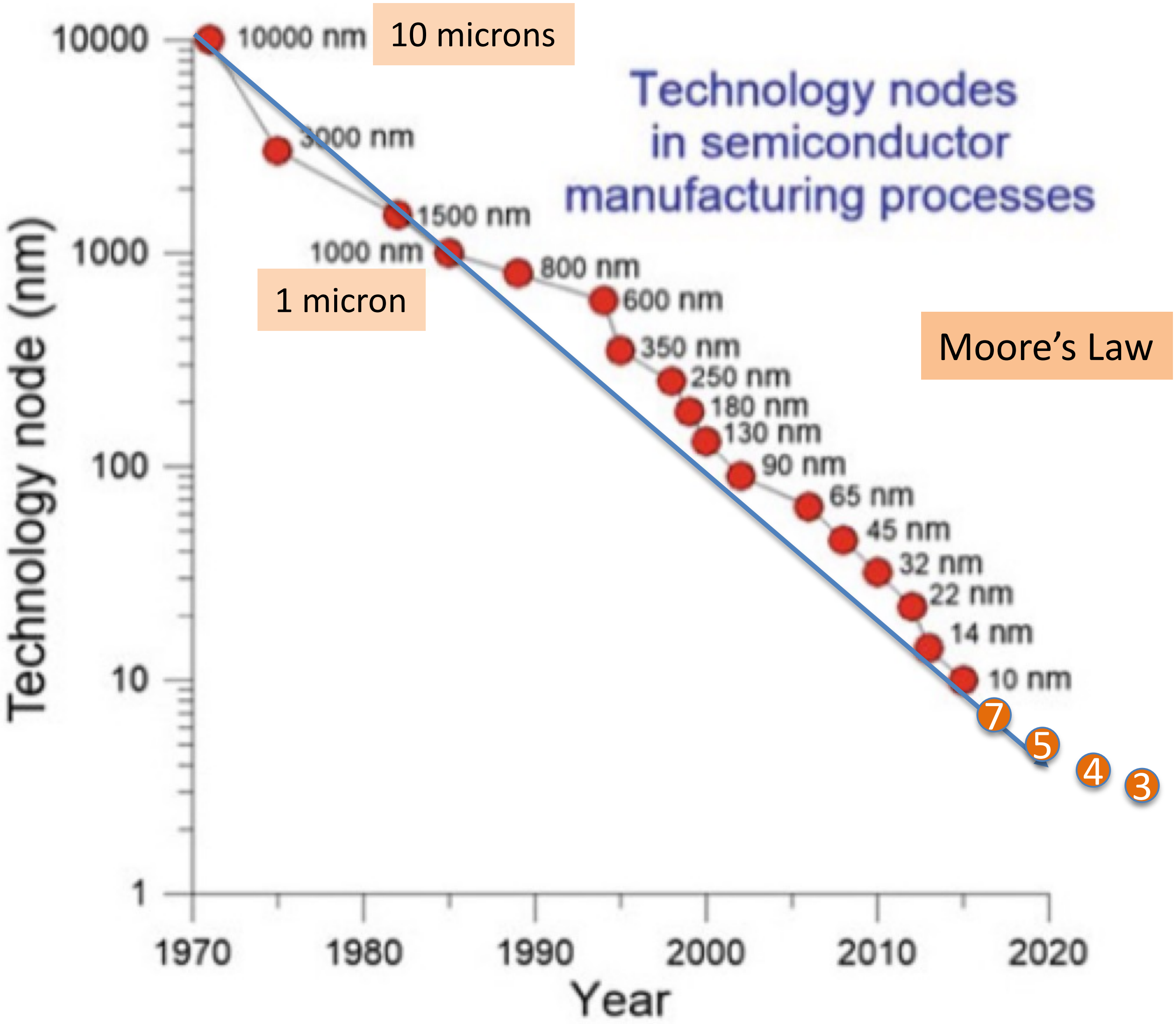


Hard disk storage has become denser at an exponential rate over the last 50 years, just like main memory. The dramatic increase in capacity and speed of both has fueled the increasing power of computers.

Node Timeline

Quora

Dave Haynie · Follow
Hardware Systems Engineer

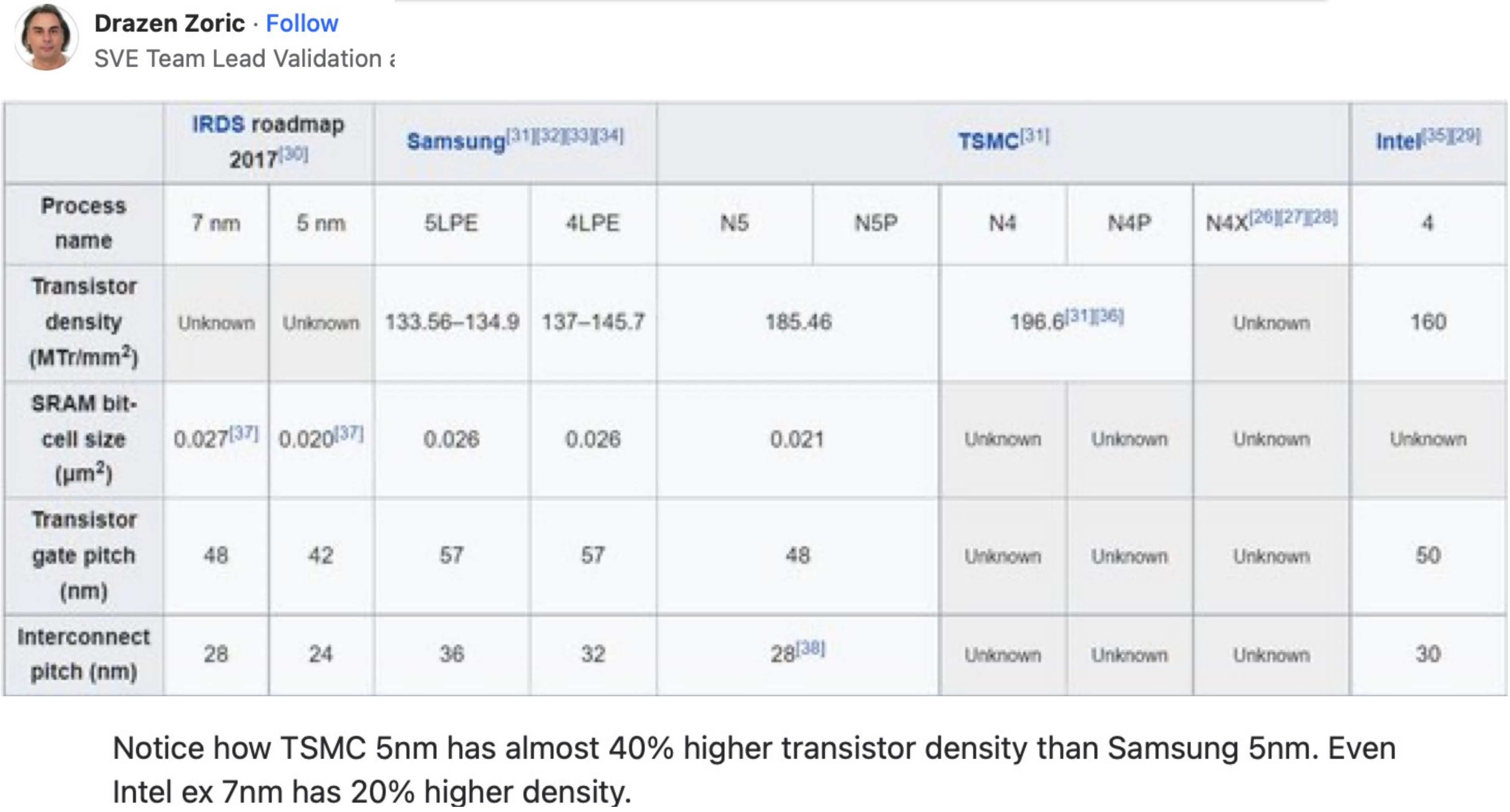


10 microns

1 micron

Moore's Law

Process Comparison

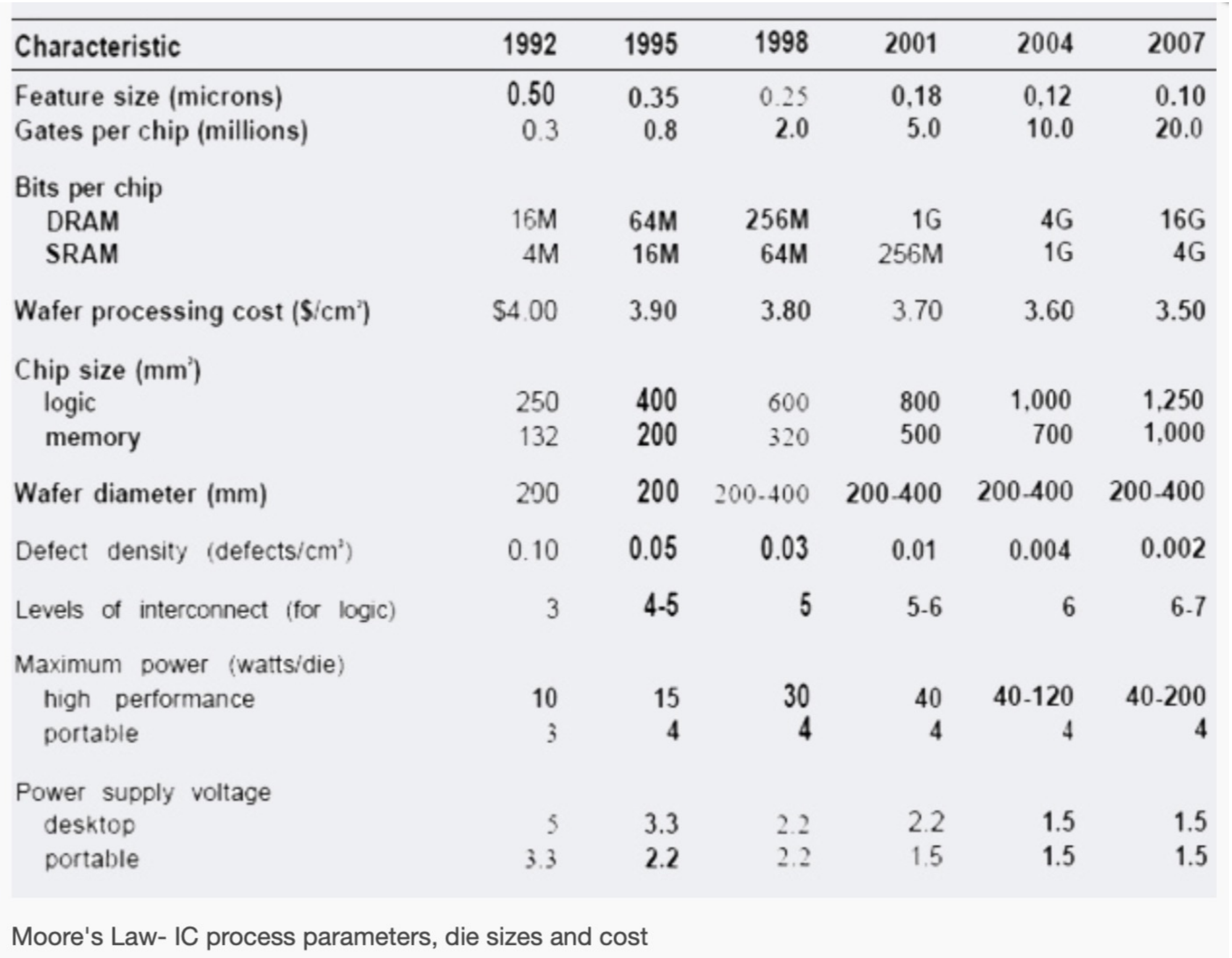


Drazen Zoric · Follow
SVE Team Lead Validation a

	IRDS roadmap 2017[30]		Samsung[31][32][33][34]		TSMC[31]					Intel[35][29]
Process name	7 nm	5 nm	5LPE	4LPE	N5	N5P	N4	N4P	N4X[26][27][28]	4
Transistor density (MTr/mm²)	Unknown	Unknown	133.56–134.9	137–145.7	185.46		196.6[31][36]		Unknown	160
SRAM bit-cell size (μm²)	0.027[37]	0.020[37]	0.026	0.026	0.021		Unknown	Unknown	Unknown	Unknown
Transistor gate pitch (nm)	48	42	57	57	48		Unknown	Unknown	Unknown	50
Interconnect pitch (nm)	28	24	36	32	28[38]		Unknown	Unknown	Unknown	30

Notice how TSMC 5nm has almost 40% higher transistor density than Samsung 5nm. Even Intel ex 7nm has 20% higher density.

Die Sizes



Characteristic	1992	1995	1998	2001	2004	2007
Feature size (microns)	0.50	0.35	0.25	0,18	0,12	0.10
Gates per chip (millions)	0.3	0.8	2.0	5.0	10.0	20.0
Bits per chip						
DRAM	16M	64M	256M	1G	4G	16G
SRAM	4M	16M	64M	256M	1G	4G
Wafer processing cost ($/cm²)	$4.00	3.90	3.80	3.70	3.60	3.50
Chip size (mm²)						
logic	250	400	600	800	1,000	1,250
memory	132	200	320	500	700	1,000
Wafer diameter (mm)	200	200	200-400	200-400	200-400	200-400
Defect density (defects/cm²)	0.10	0.05	0.03	0.01	0.004	0.002
Levels of interconnect (for logic)	3	4-5	5	5-6	6	6-7
Maximum power (watts/die)						
high performance	10	15	30	40	40-120	40-200
portable	3	4	4	4	4	4
Power supply voltage						
desktop	5	3.3	2.2	2.2	1.5	1.5
portable	3.3	2.2	2.2	1.5	1.5	1.5

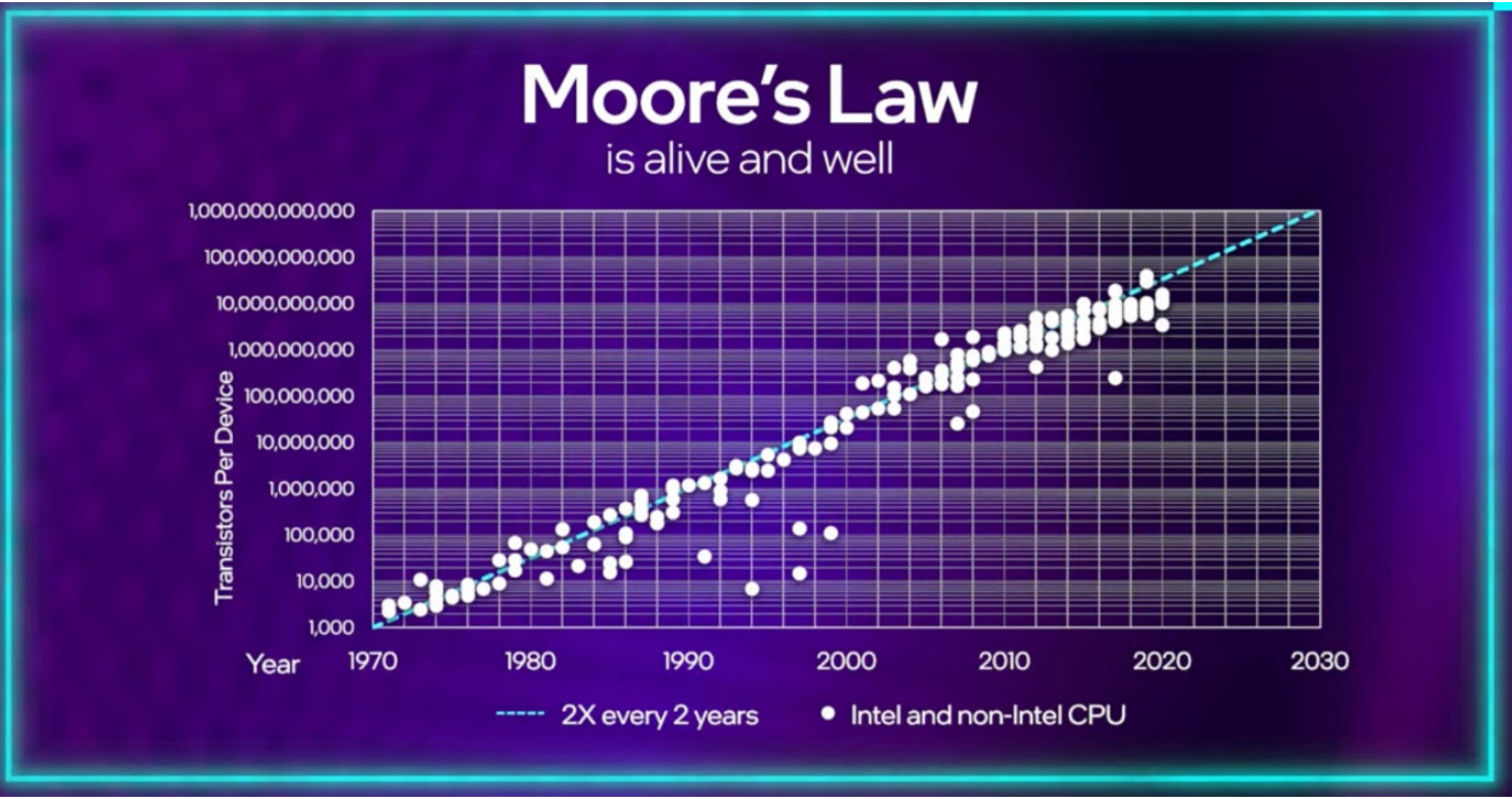
Moore's Law- IC process parameters, die sizes and cost

Intel’s View of Moore

By **Intel**

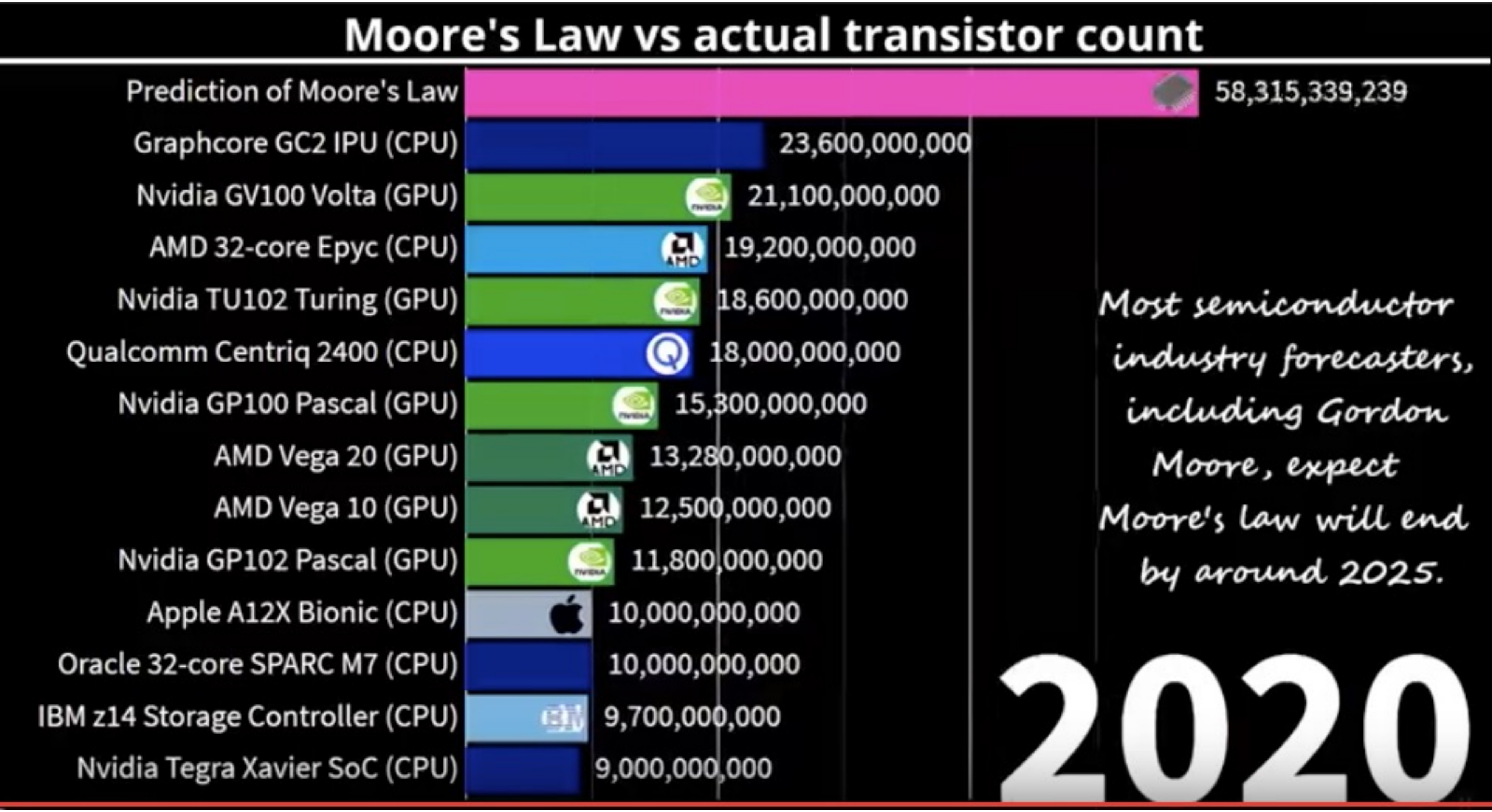
Nov 2021

Intel CEO Gelsinger just claimed Intel will exceed this rate!



Moore’s Law 2020

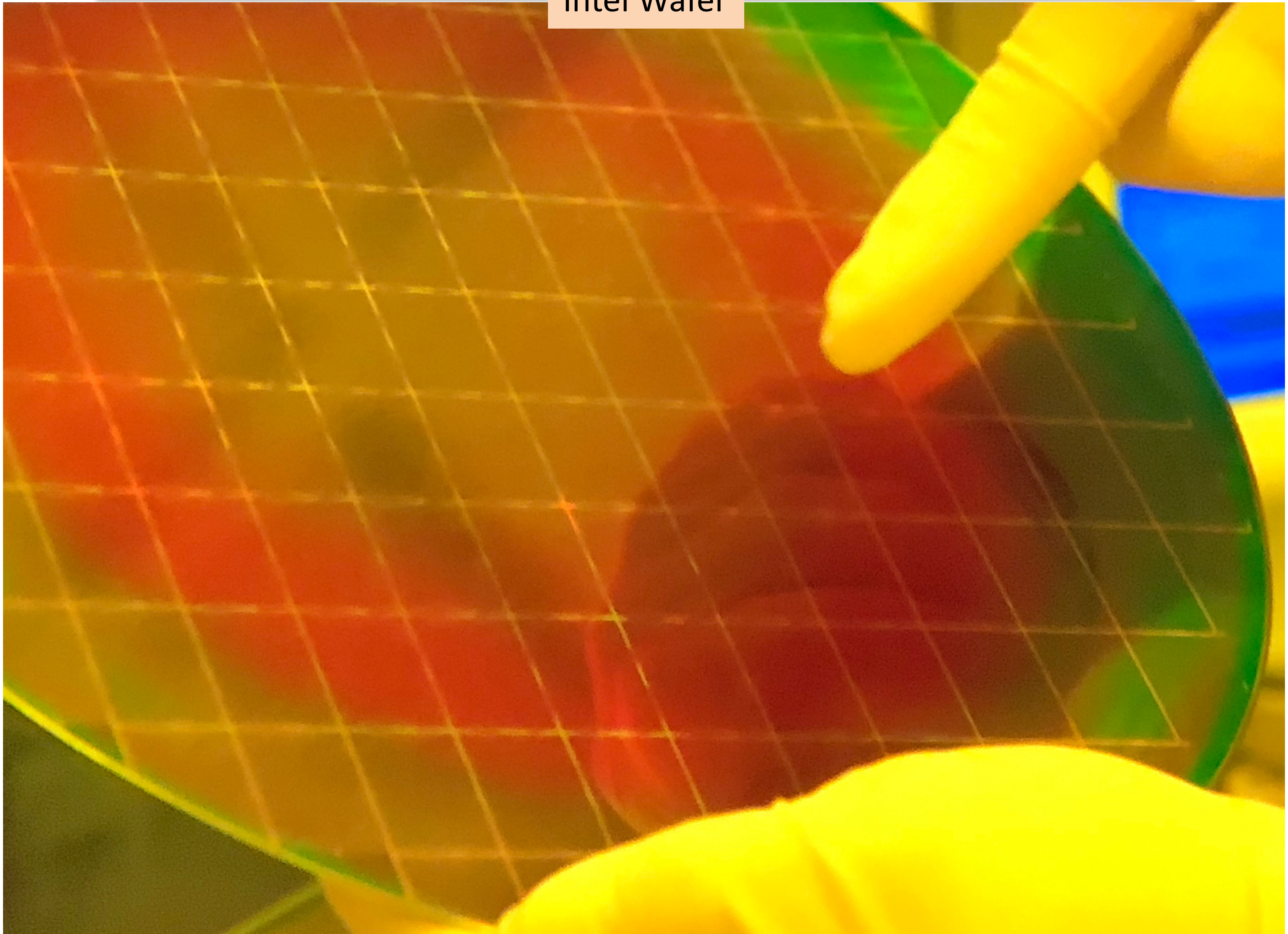
By **TSMC**



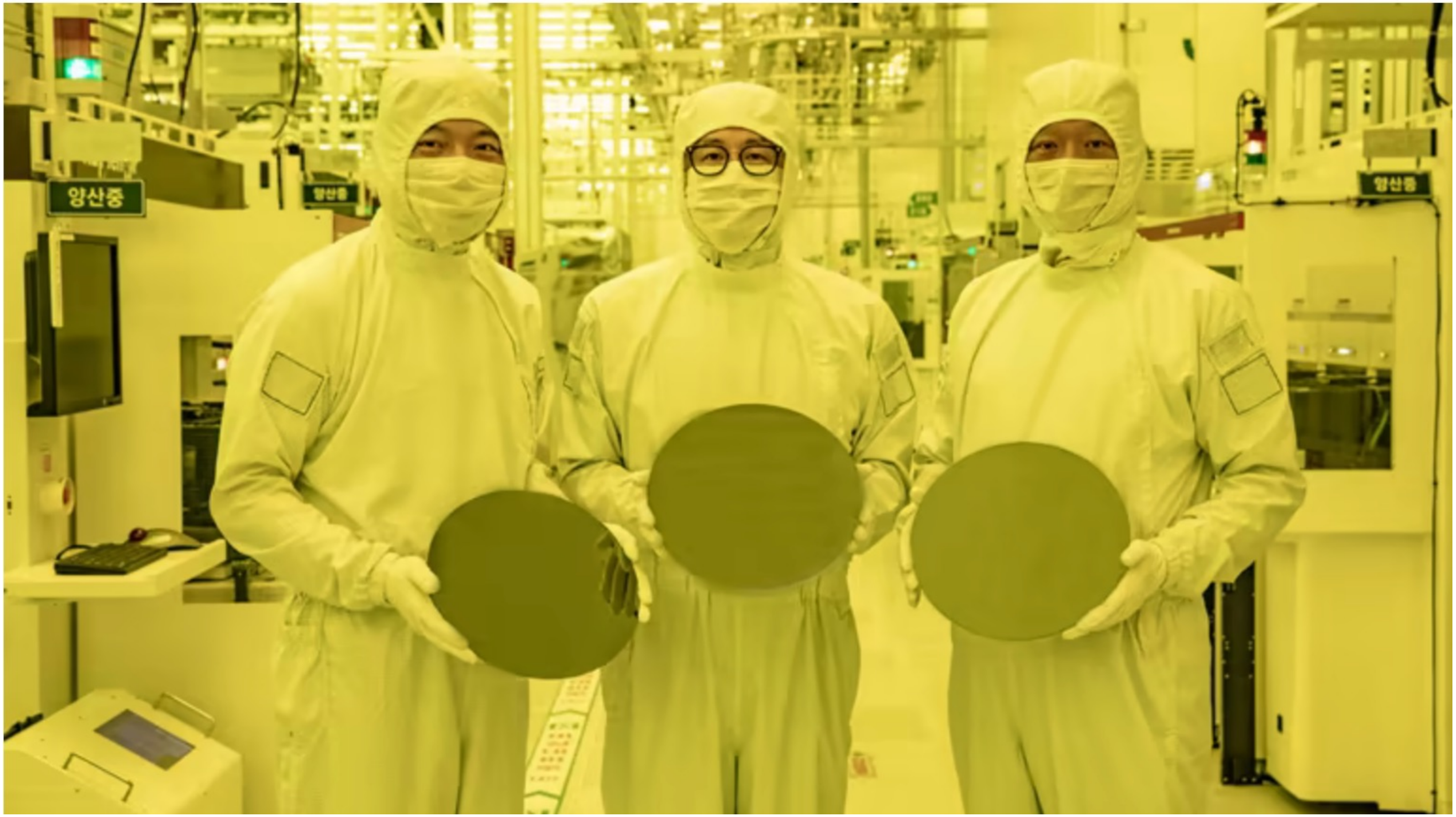
Most semiconductor industry forecasters, including Gordon Moore, expect Moore's law will end by around 2025.

Chip Shortage

Intel Wafer

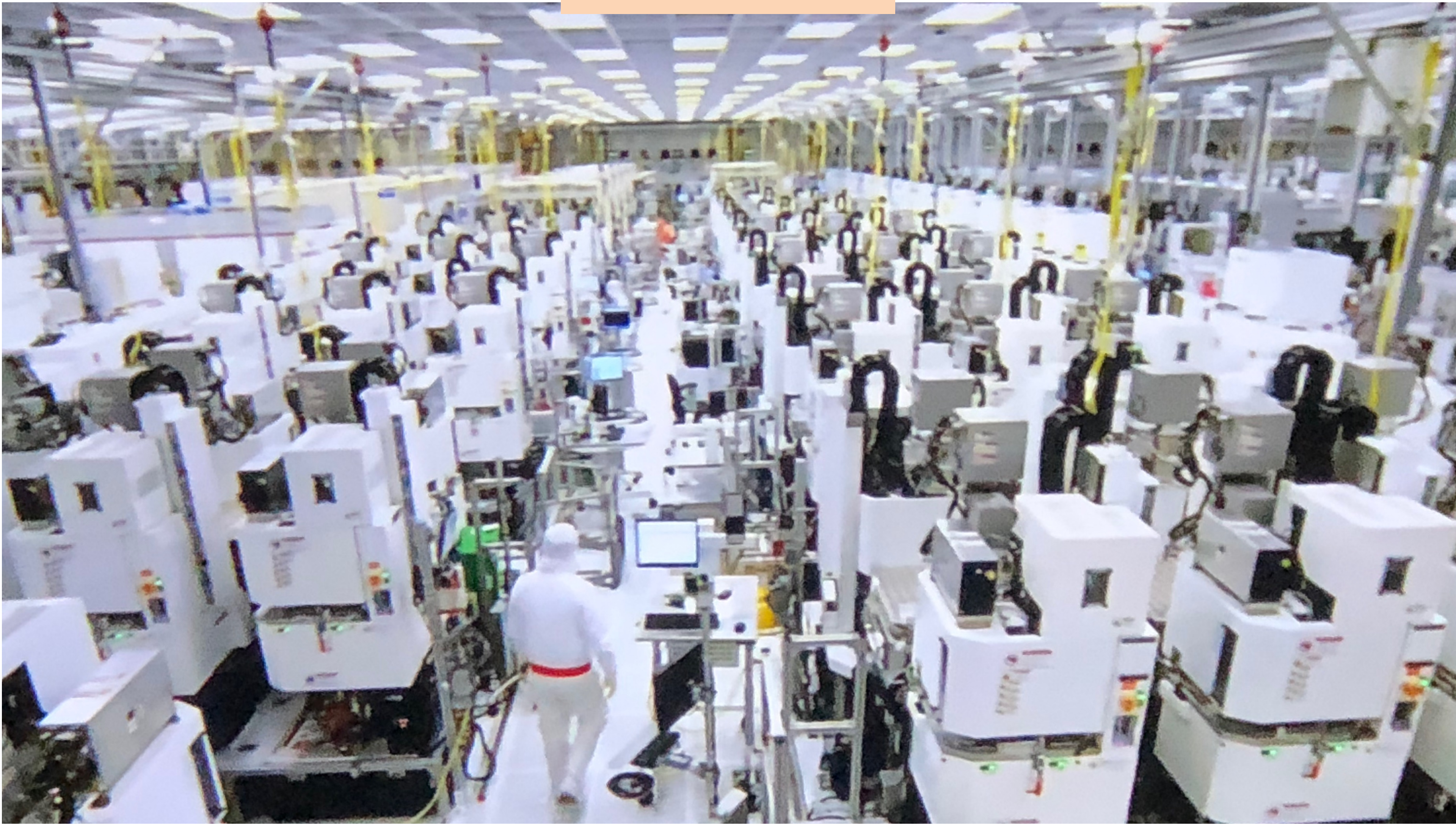


Samsung 12” Wafers



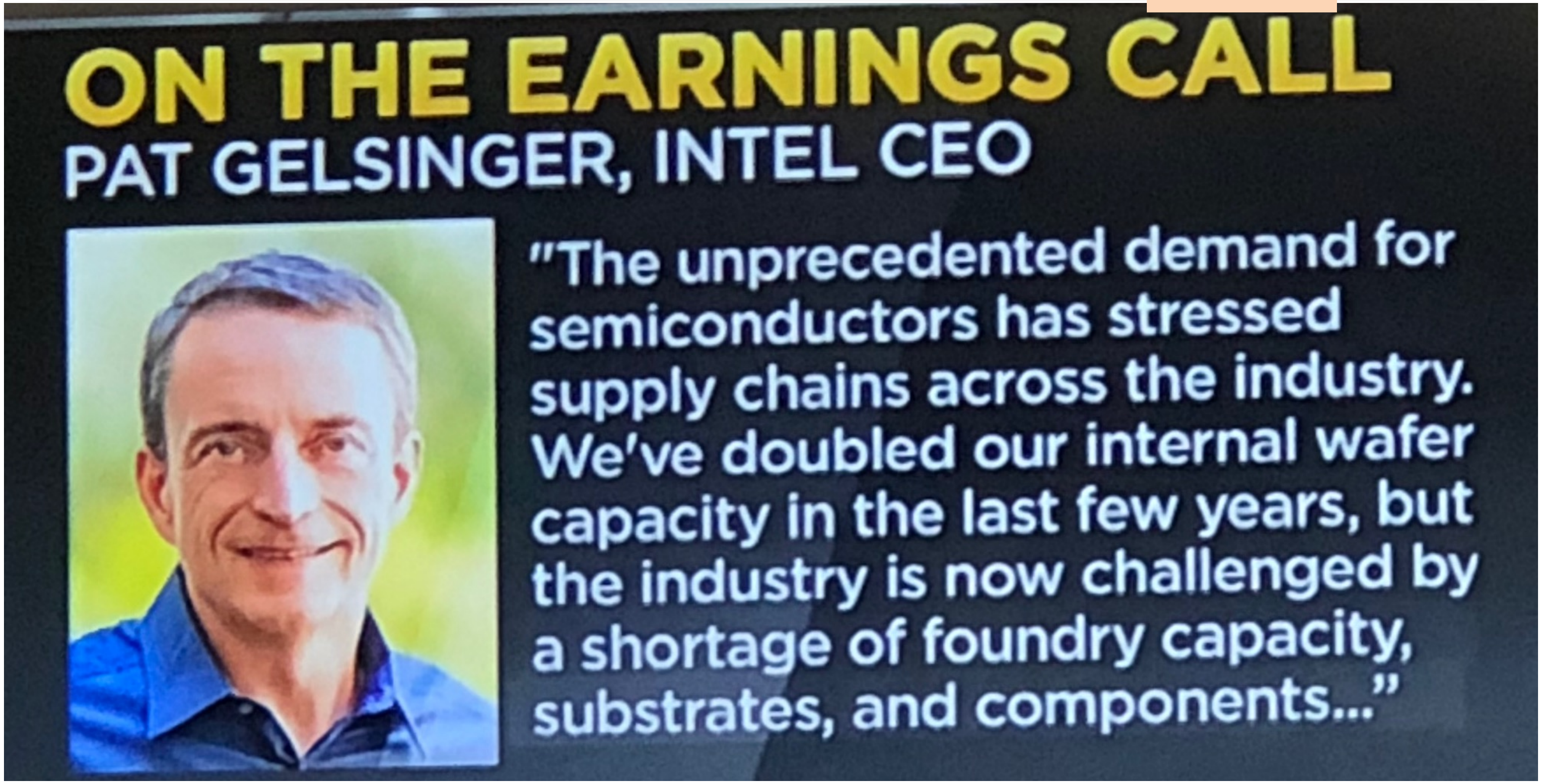
Chip Shortage

Intel Automated Fab



Intel CEO on Chip Shortage

4-23-21

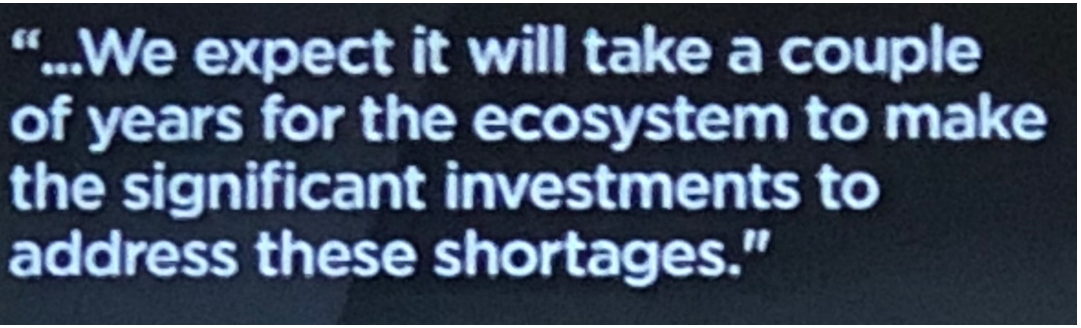


ON THE EARNINGS CALL

PAT GELSINGER, INTEL CEO

"The unprecedented demand for semiconductors has stressed supply chains across the industry. We've doubled our internal wafer capacity in the last few years, but the industry is now challenged by a shortage of foundry capacity, substrates, and components..."

Intel building 2 new fabs in AZ **$20B**



"...We expect it will take a couple of years for the ecosystem to make the significant investments to address these shortages."

- ❖ Intel, Micron only remaining US fabs
- ❖ Intel produces **17%** of world supply
- ❖ US produced **37%** 20 yrs ago
- ❖ AMD divested its 25 fabs in 2006

Chip Shortage

Intel New Fabs in Phoenix

Intel Foundry for MediaTek

Intel, MediaTek Enter Into Chip-Manufacturing Agreement

5:50 AM ET, 07/25/2022 - MT Newswires05:50 AM EDT, 07/25/2022 (MT Newswires) -- Intel (INTC) said Monday it entered into an agreement with MediaTek to manufacture chips using Intel Foundry Services.

The chipmaker said MediaTek aims to use Intel's process technologies to produce multiple chips for *smart edge* devices.

Financial details of the agreement were not disclosed.

Intel’s Newest Fabs

$**20B** in Ohio

Online end of **2025**

SATURDAY, JANUARY 22, 2022 A9

Intel to build chip factories in Ohio

Company will invest $20 billion as a global shortage highlights the risks of reliance on manufacturers in Asia.

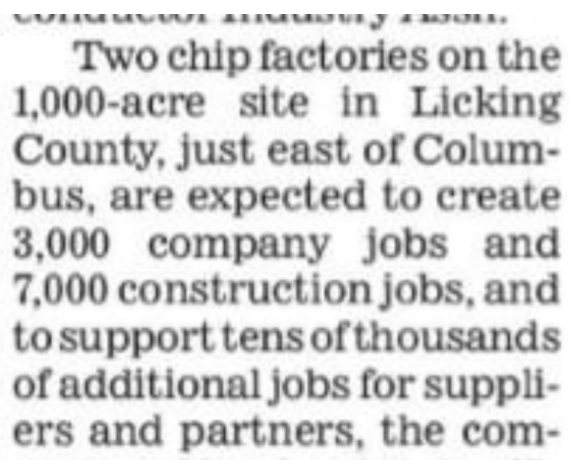
Samsung in Texas

Chipmakers are diversifying their manufacturing sites in response to the shortages. Samsung said in November that it planned to build a $17-billion factory outside Austin, Texas.

Micron Technology, based in Boise, Idaho, said it would invest $150 billion globally over the next decade in developing its line of memory chips, with a potential U.S. manufacturing expansion if tax credits can help make up for the higher costs of American manufacturing.

Micron globally

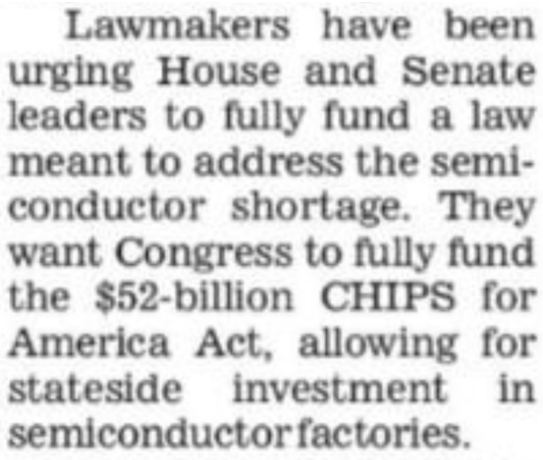
10,000 jobs in Ohio



conductor Industry Assn.

Two chip factories on the 1,000-acre site in Licking County, just east of Columbus, are expected to create 3,000 company jobs and 7,000 construction jobs, and to support tens of thousands of additional jobs for suppliers and partners, the com-

CHIPS for America Act

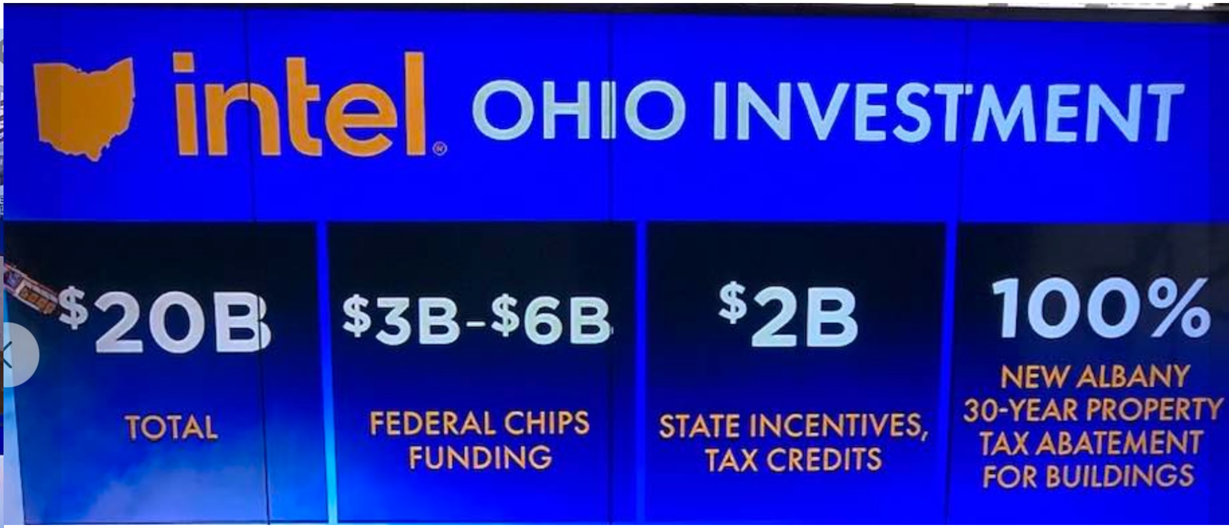


Lawmakers have been urging House and Senate leaders to fully fund a law meant to address the semiconductor shortage. They want Congress to fully fund the $52-billion CHIPS for America Act, allowing for stateside investment in semiconductor factories.

CHIPS Act in Ohio

Sept 2022

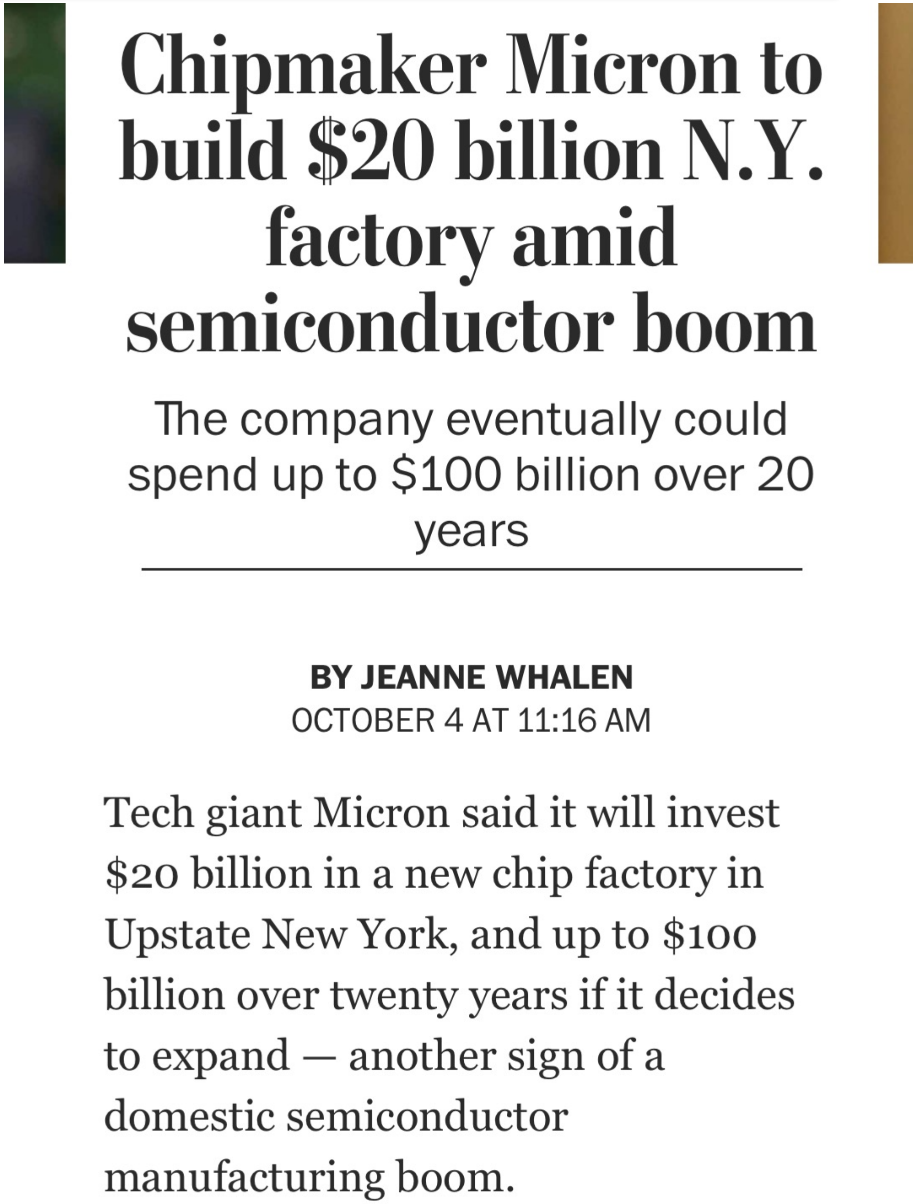
PRES. BIDEN: THE FUTURE OF THE CHIP INDUSTRY WILL BE MADE IN AMERICA


intel OHIO INVESTMENT

$20B	$3B-$6B	$2B	100%
TOTAL	FEDERAL CHIPS FUNDING	STATE INCENTIVES, TAX CREDITS	NEW ALBANY 30-YEAR PROPERTY TAX ABATEMENT FOR BUILDINGS

Micron in NY

Chipmaker Micron to build $20 billion N.Y. factory amid semiconductor boom



Chipmaker Micron to build $20 billion N.Y. factory amid semiconductor boom

The company eventually could spend up to $100 billion over 20 years

BY JEANNE WHALEN
OCTOBER 4 AT 11:16 AM

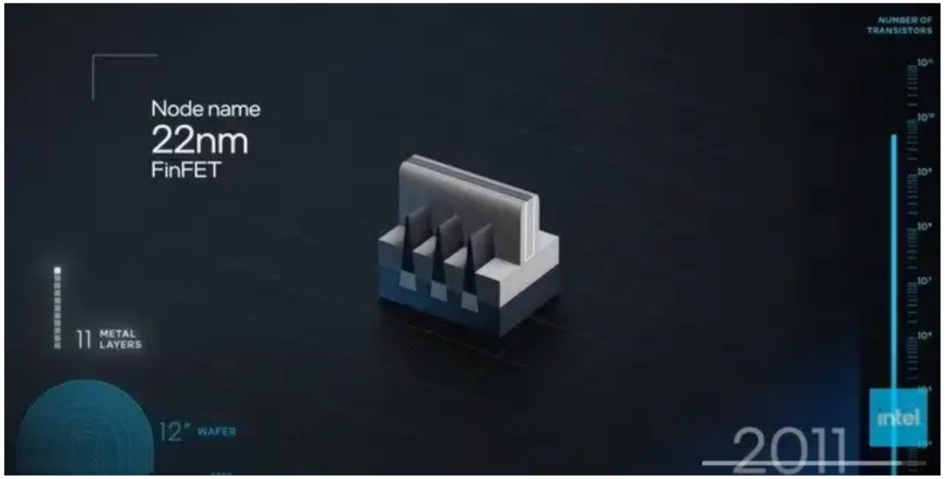
Tech giant Micron said it will invest $20 billion in a new chip factory in Upstate New York, and up to $100 billion over twenty years if it decides to expand — another sign of a domestic semiconductor manufacturing boom.

Intel MOSFET

Intel video

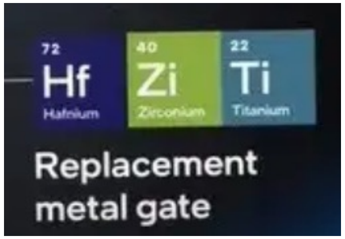
https://www.youtube.com/watch?v=Z7M8etXUEUU&t=47s

FinFET: 2011

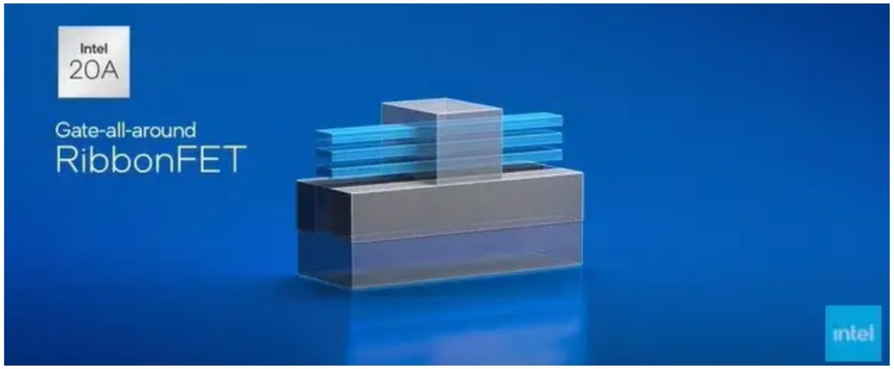


Intel amazed the industry with its aggressive adoption of a new transistor topology at the 22nm process node – the FinFET (also known as the "tri-gate FET").

SiON
Gate Dielectric
14
Si
Silicon
8
O
Oxygen
7
N
Nitrogen



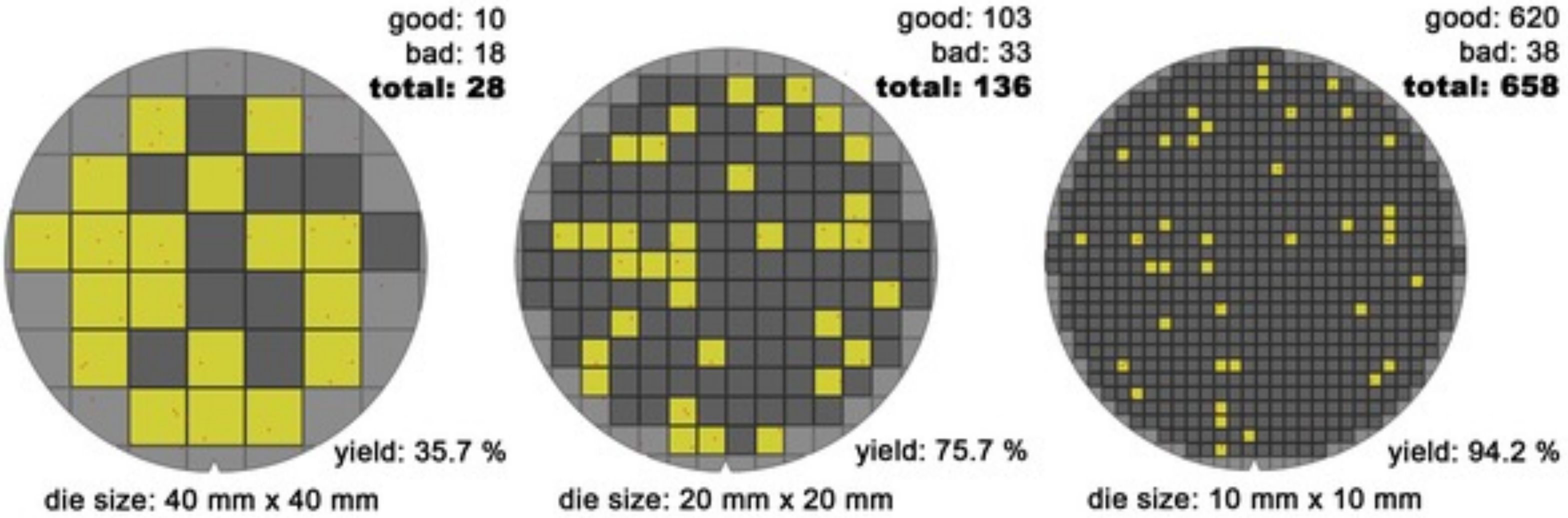
Gate-All Around (GAA) Ribbon FET: Intel 20A in 2024



To further improve the electrostatic gate control over the channel, another major evolution in the transistor topology is emerging to replace the FinFET. A gate-all-around configuration involves a vertical stack of electrically isolated silicon channels. The gate dielectric and gate input utilize an atomic layer deposition (ALD) process flow to surround all channel surfaces in the stack.

Intel will be releasing their GAA *Ribbon FET* 20A process in 1H 2024.

Wafers: Yield

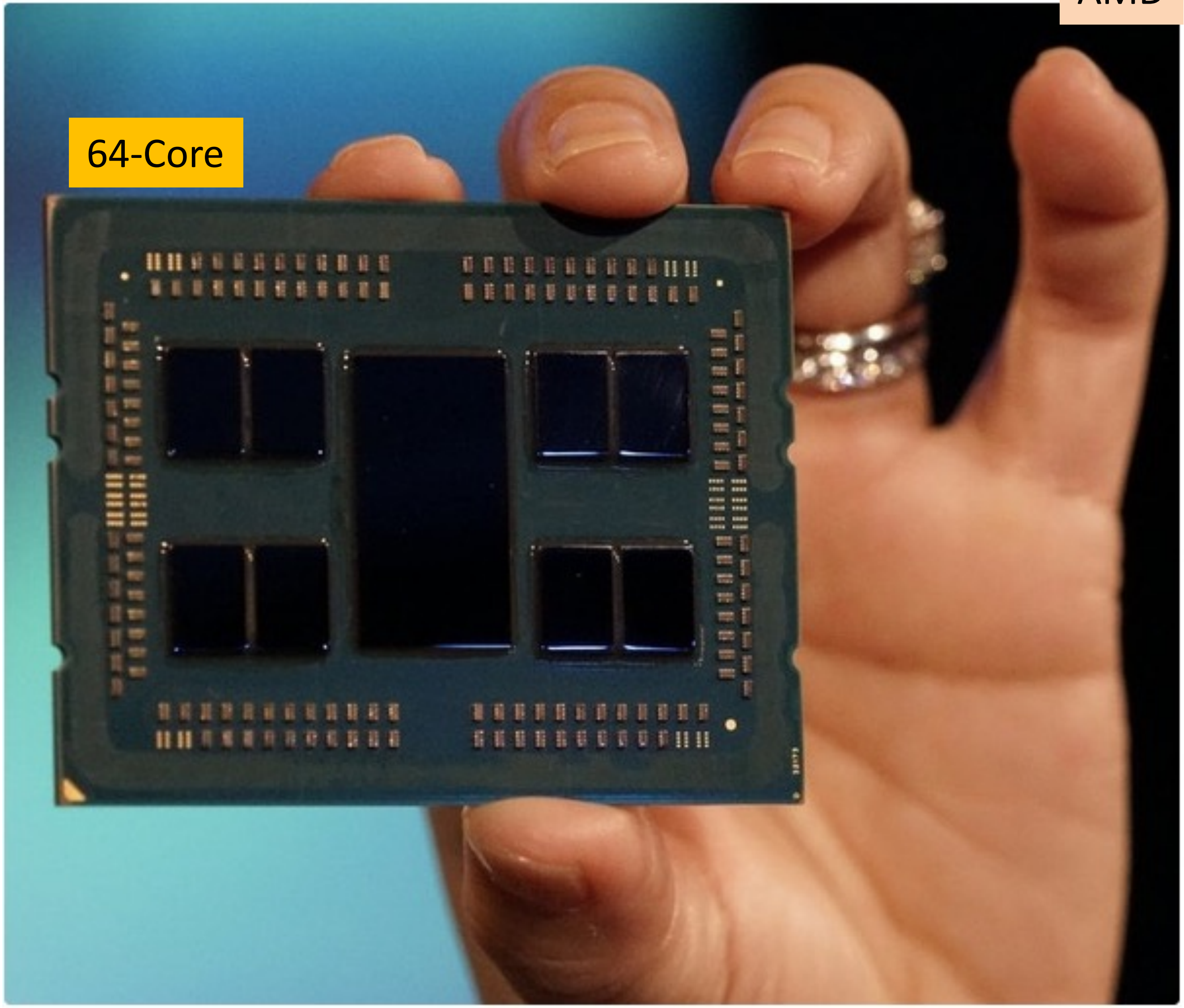


yellow **shows bad dice** (a function of defect density)

Section

Multi-Core

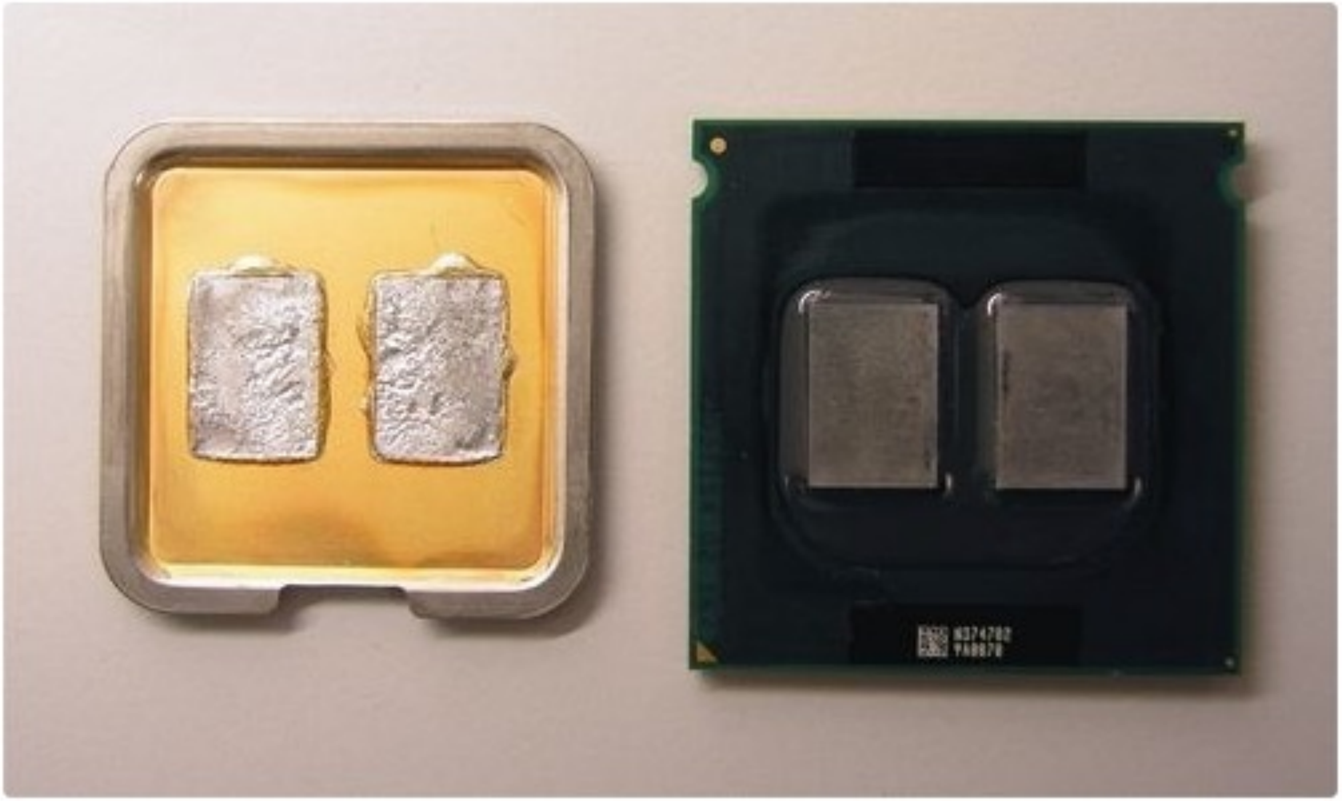
Multi-Cores + L2/L3



Lisa Su proudly shows off her 64-core EPYC monster at CES.

Multi-Cores + L2/L3

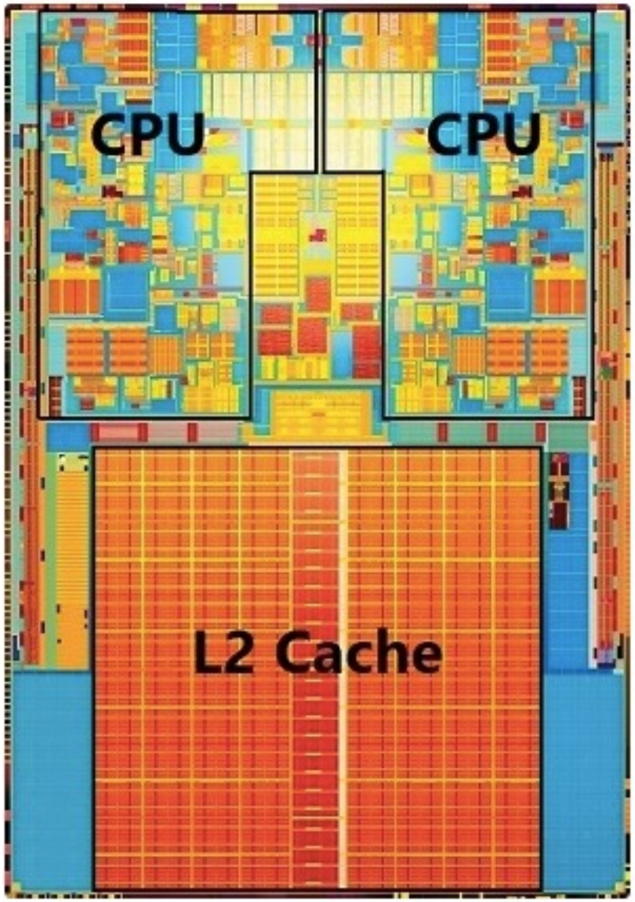
Intel

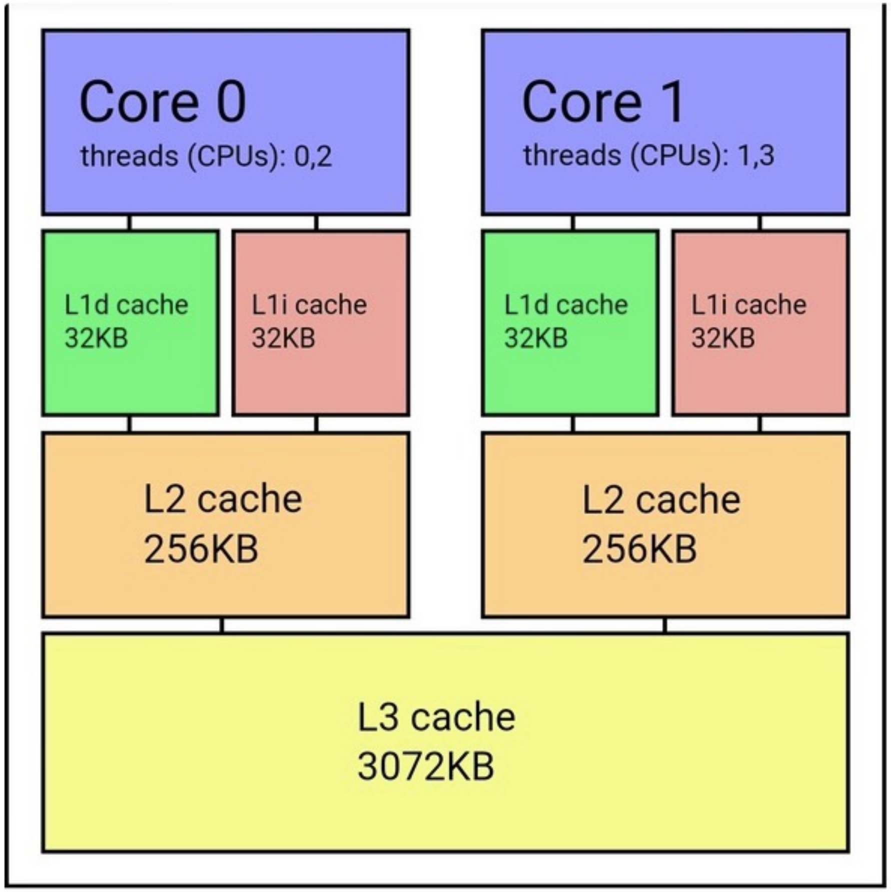


This is a delided Core 2 Quad with the two Core 2 Duo dies.

Multi-Cores + L2/L3

Intel

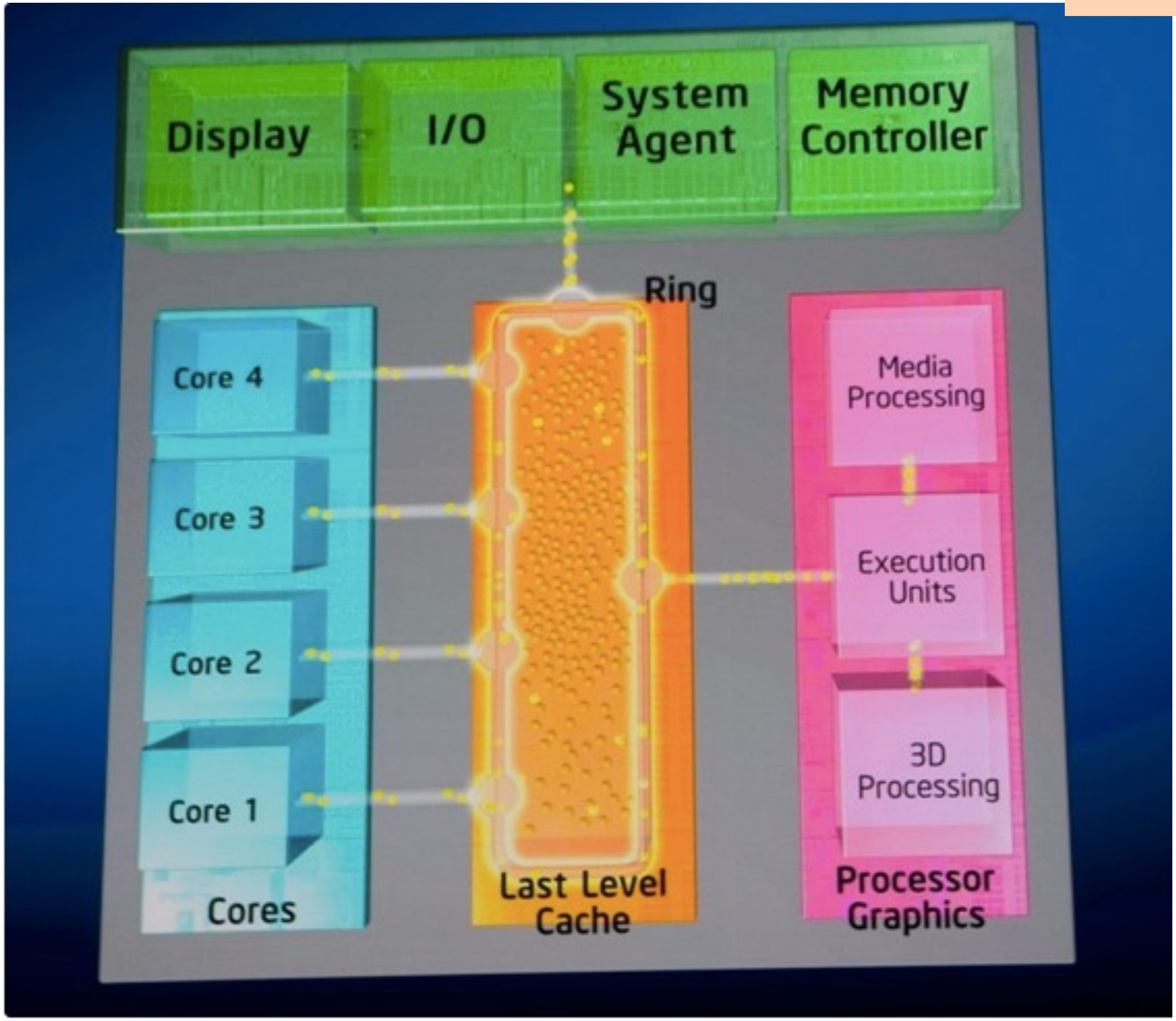




This is a late Core 2 Duo (Wolfdale), note how massive the uncore L2 is compared to the actual CPU cores. (uncore = outside the CPU)

Multi-Cores + L2/L3

Intel

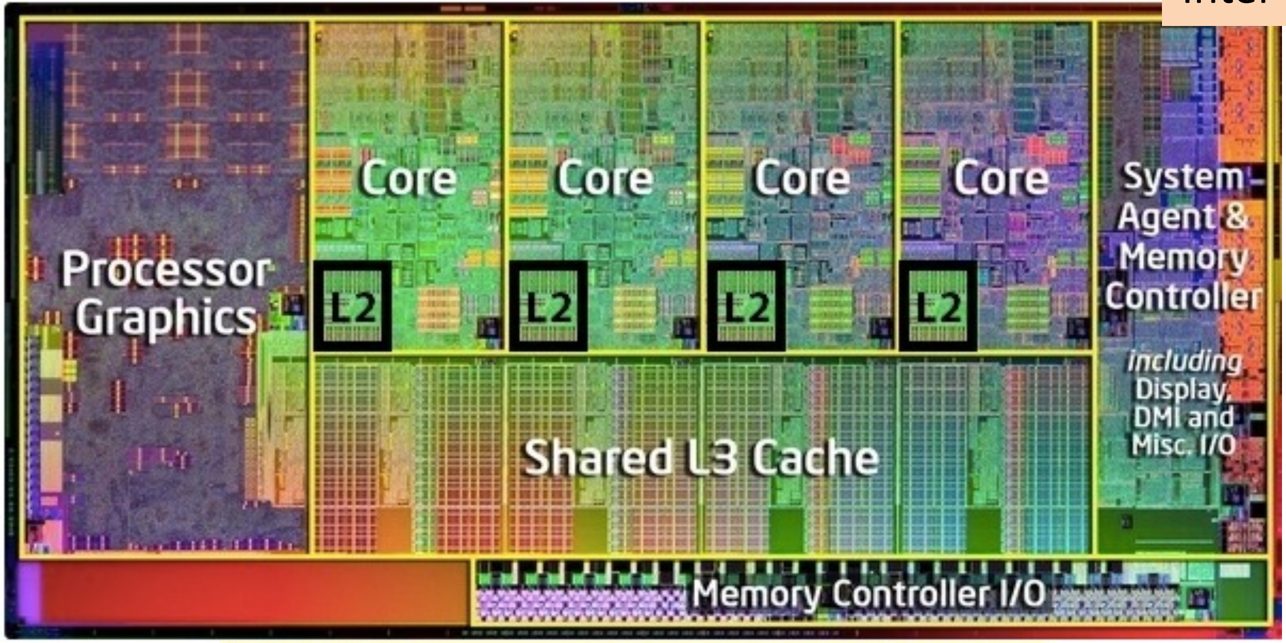


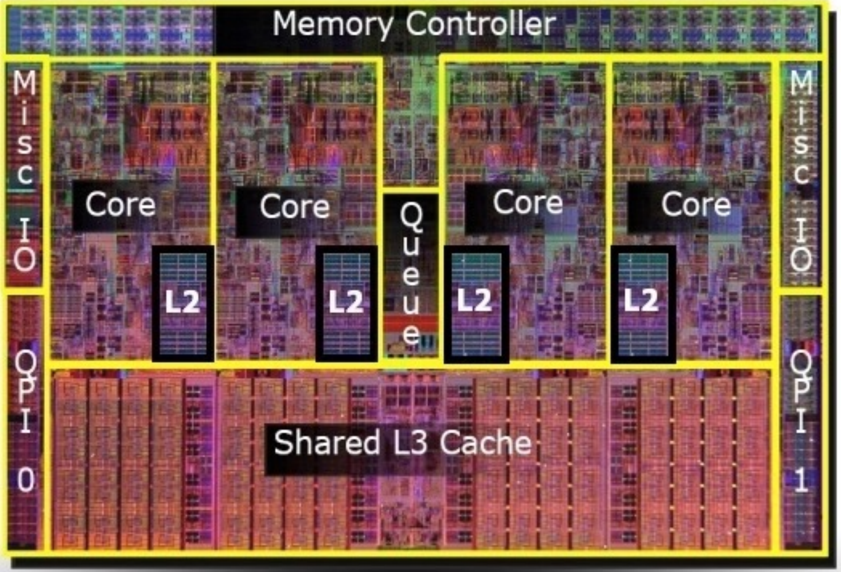
This special 'glue' remains in use today in all non-HEDT Intel CPUs.

Multi-Cores + L2/L3

Intel

Core 4

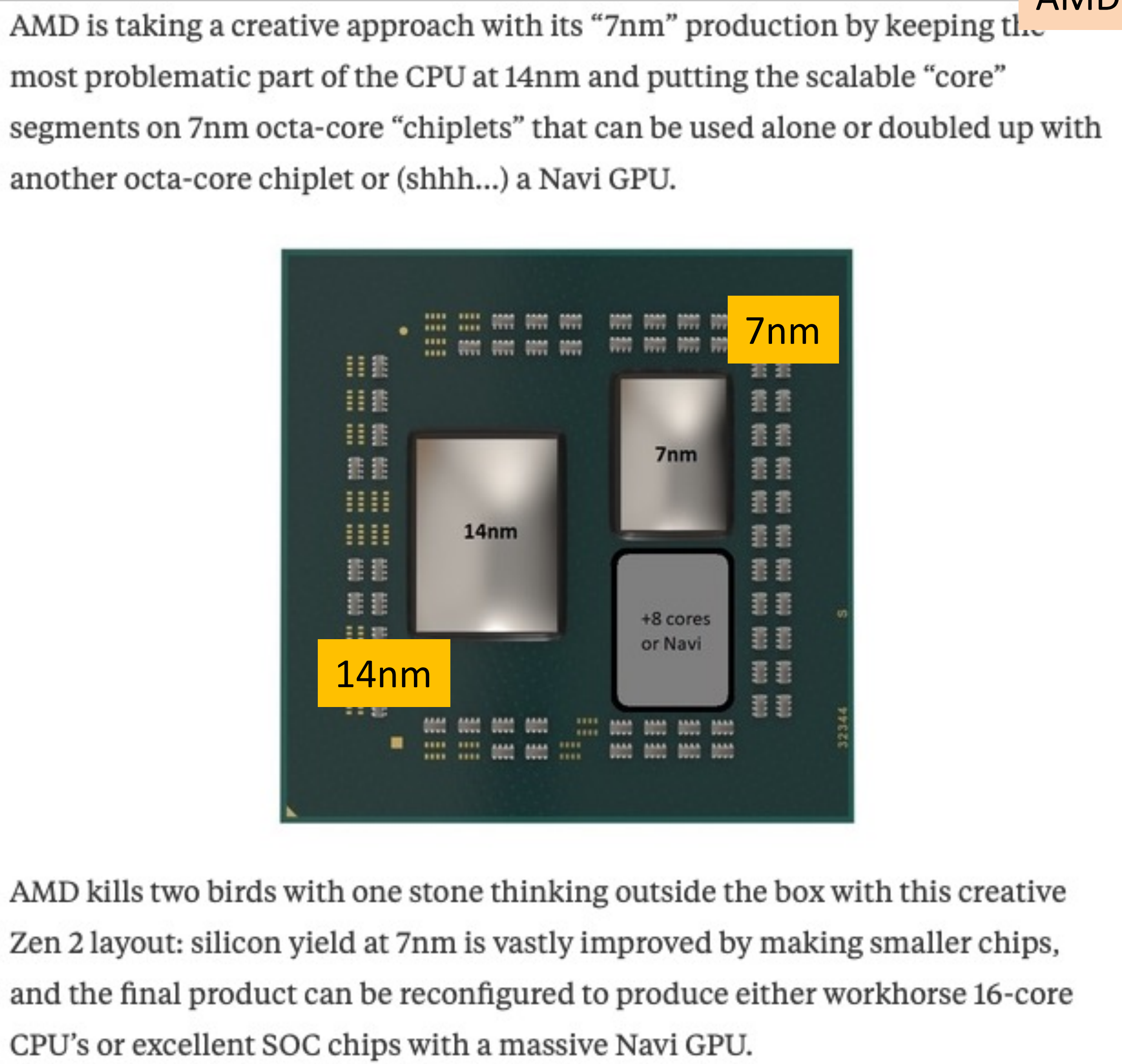




Multi-Cores + L2/L3

AMD

Zen 2

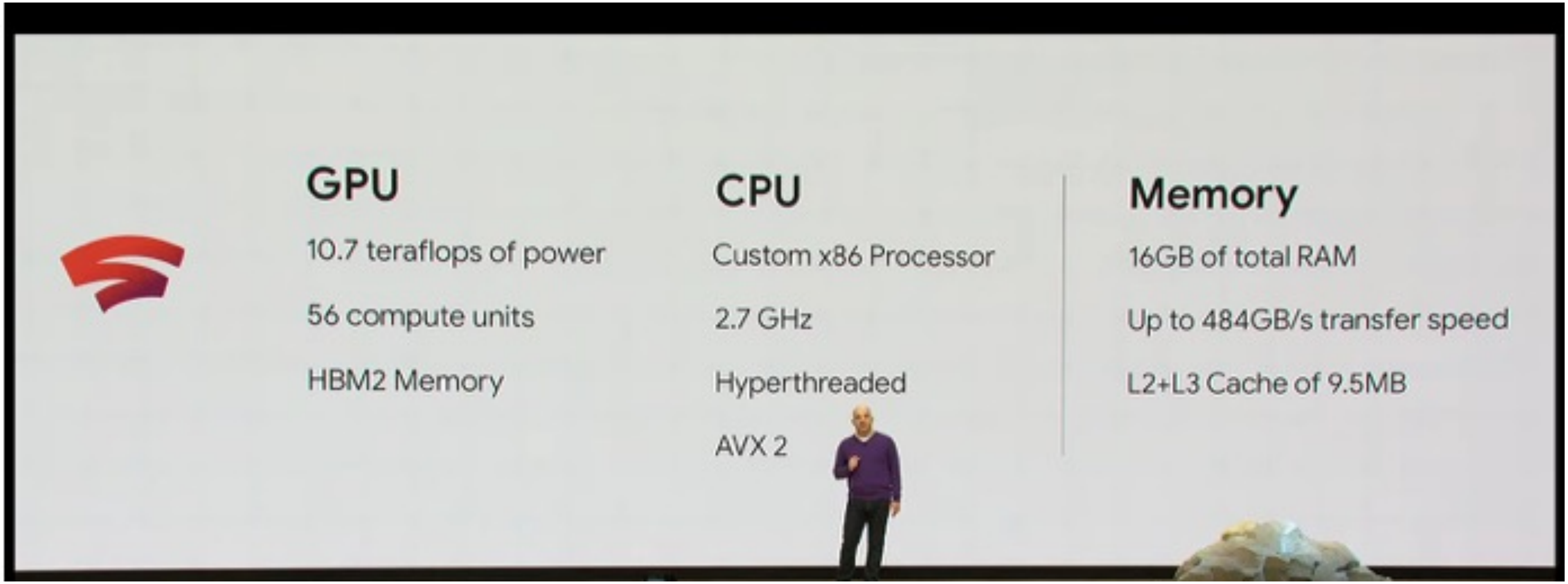


AMD is taking a creative approach with its "7nm" production by keeping the most problematic part of the CPU at 14nm and putting the scalable "core" segments on 7nm octa-core "chiplets" that can be used alone or doubled up with another octa-core chiplet or (shhh...) a Navi GPU.

AMD kills two birds with one stone thinking outside the box with this creative Zen 2 layout: silicon yield at 7nm is vastly improved by making smaller chips, and the final product can be reconfigured to produce either workhorse 16-core CPU's or excellent SOC chips with a massive Navi GPU.

Multi-Cores + L2/L3

AMD



GPU

10.7 teraflops of power

56 compute units

HBM2 Memory

CPU

Custom x86 Processor

2.7 GHz

Hyperthreaded

AVX 2

Memory

16GB of total RAM

Up to 484GB/s transfer speed

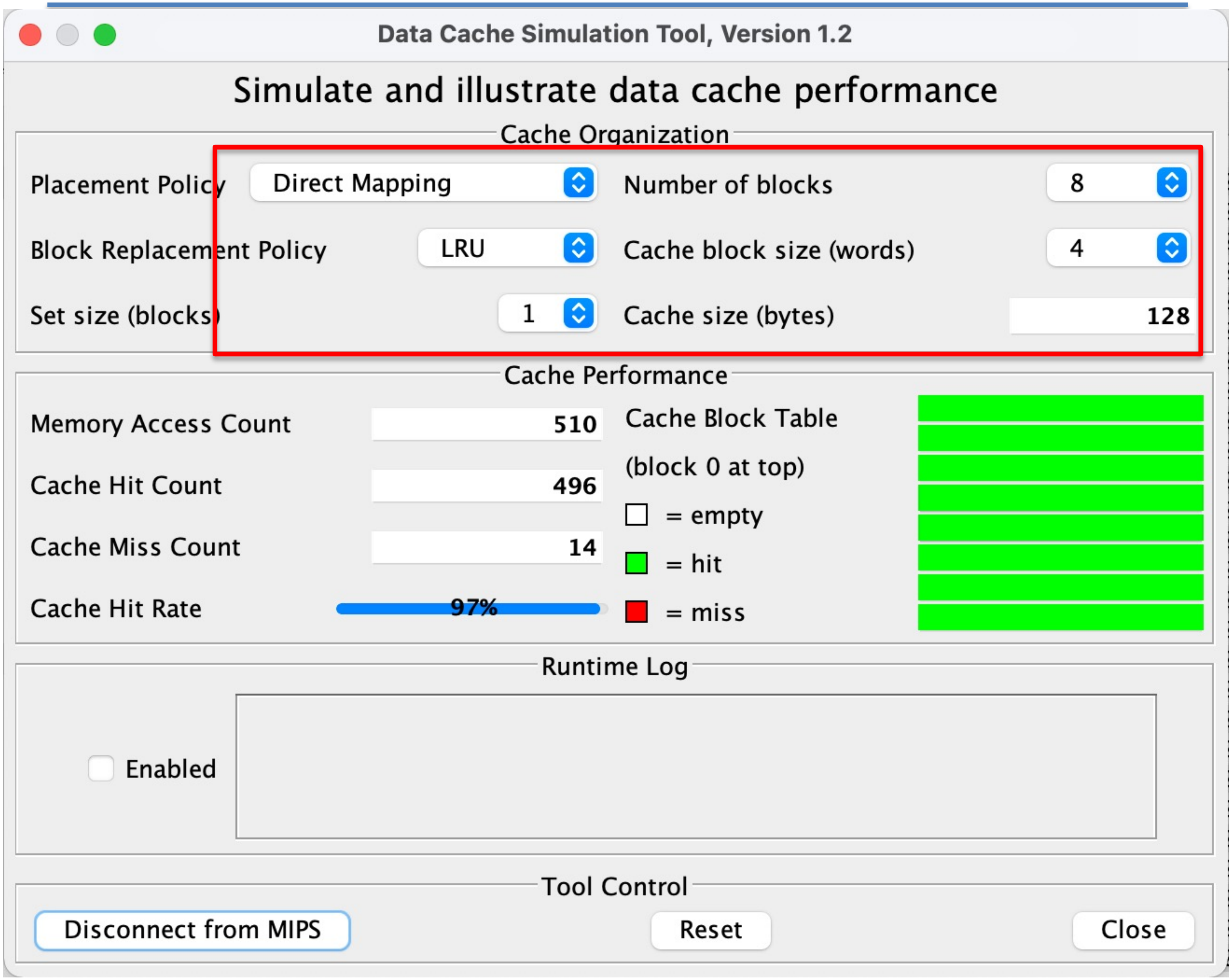
L2+L3 Cache of 9.5MB

Section

MLM:
Caches

See slides in 122 Part 1: Memory

Cache Simulation



Data Cache Simulation Tool, Version 1.2

Simulate and illustrate data cache performance

Cache Organization

Placement Policy: Direct Mapping
Block Replacement Policy: LRU
Set size (blocks): 1
Number of blocks: 8
Cache block size (words): 4
Cache size (bytes): 128

Cache Performance

Memory Access Count: 510
Cache Hit Count: 496
Cache Miss Count: 14
Cache Hit Rate: 97%

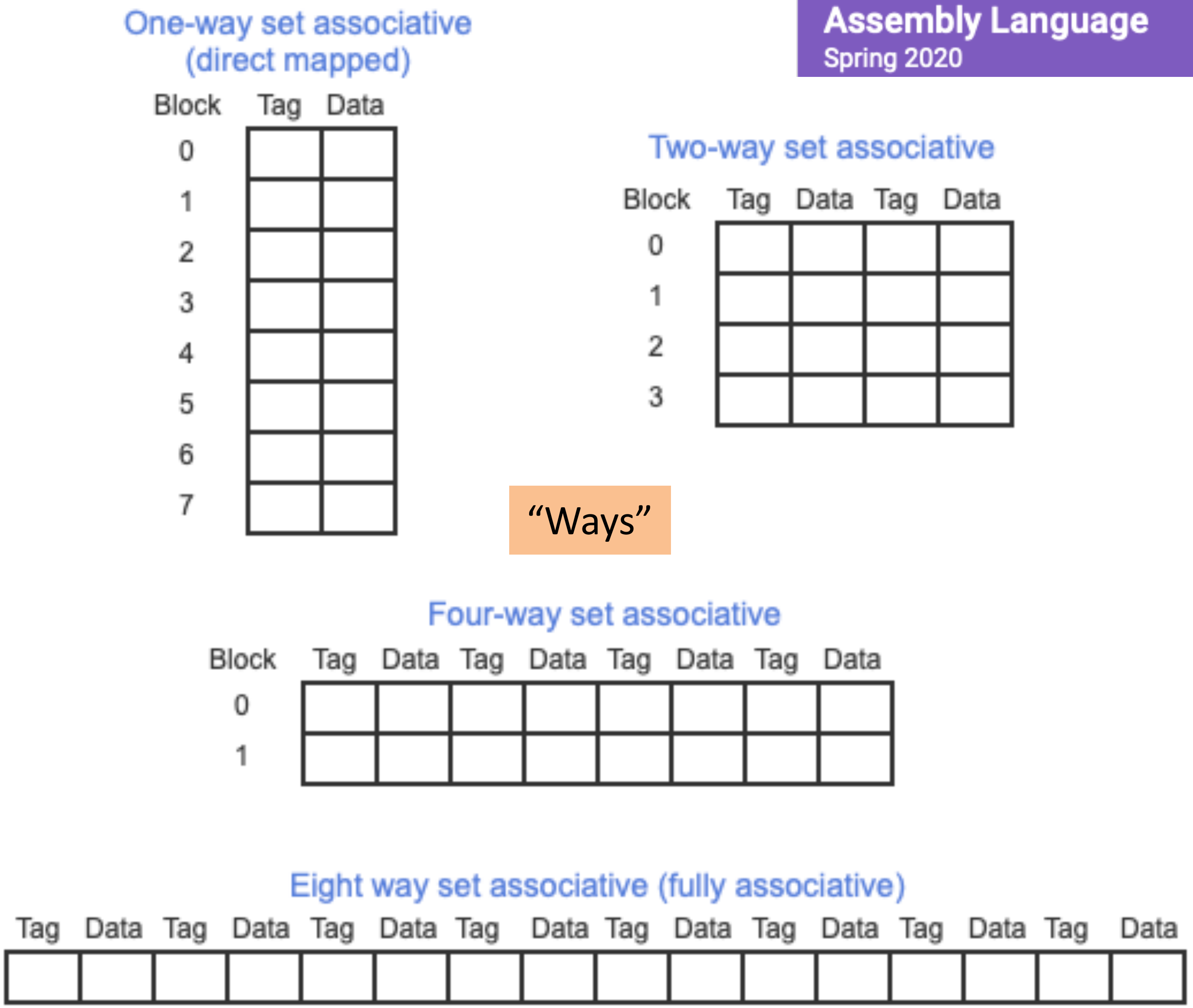
Cache Block Table (block 0 at top)
□ = empty
🟩 = hit
🟥 = miss

Runtime Log

□ Enabled

Tool Control

Cache Memory



One-way set associative
(direct mapped)

Block	Tag	Data
0		
1		
2		
3		
4		
5		
6		
7		

Two-way set associative

Block	Tag	Data	Tag	Data
0				
1				
2				
3				

“Ways”

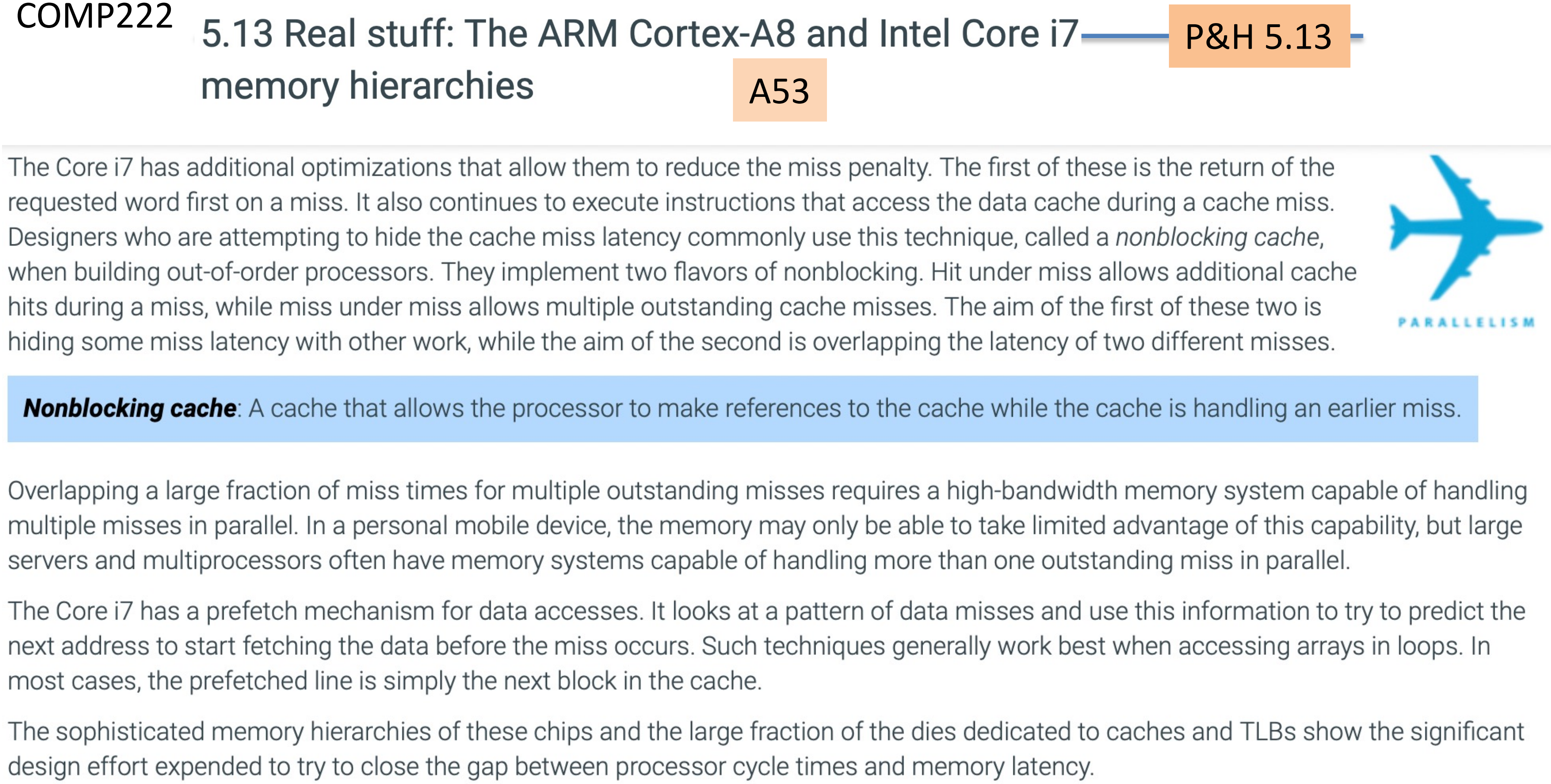
Four-way set associative

Block	Tag	Data	Tag	Data	Tag	Data	Tag	Data
0								
1								

Eight way set associative (fully associative)

Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data

Cache Data



5.13 Real stuff: The ARM Cortex-A8 and Intel Core i7 memory hierarchies

P&H 5.13

A53

The Core i7 has additional optimizations that allow them to reduce the miss penalty. The first of these is the return of the requested word first on a miss. It also continues to execute instructions that access the data cache during a cache miss. Designers who are attempting to hide the cache miss latency commonly use this technique, called a *nonblocking cache*, when building out-of-order processors. They implement two flavors of nonblocking. Hit under miss allows additional cache hits during a miss, while miss under miss allows multiple outstanding cache misses. The aim of the first of these two is hiding some miss latency with other work, while the aim of the second is overlapping the latency of two different misses.

Nonblocking cache: A cache that allows the processor to make references to the cache while the cache is handling an earlier miss.

Overlapping a large fraction of miss times for multiple outstanding misses requires a high-bandwidth memory system capable of handling multiple misses in parallel. In a personal mobile device, the memory may only be able to take limited advantage of this capability, but large servers and multiprocessors often have memory systems capable of handling more than one outstanding miss in parallel.

The Core i7 has a prefetch mechanism for data accesses. It looks at a pattern of data misses and use this information to try to predict the next address to start fetching the data before the miss occurs. Such techniques generally work best when accessing arrays in loops. In most cases, the prefetched line is simply the next block in the cache.

The sophisticated memory hierarchies of these chips and the large fraction of the dies dedicated to caches and TLBs show the significant design effort expended to try to close the gap between processor cycle times and memory latency.

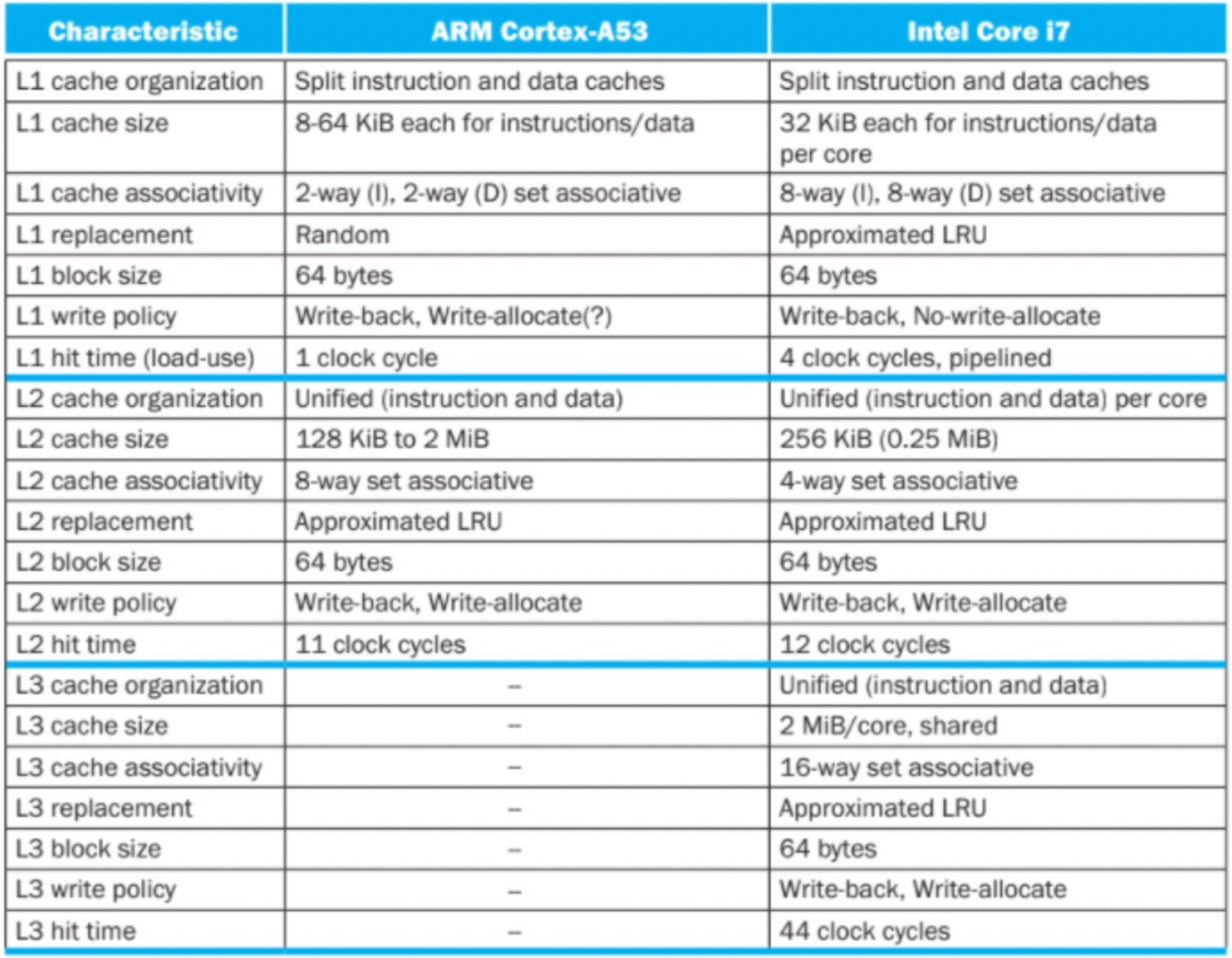
Cache Data

5.13 Real stuff: The ARM Cortex-A8 and Intel Core i7 — P&H 5.13

A53

Figure 5.13.2: Caches in the ARM Cortex-A53 and Intel Core i7 6700 (COD Figure 5.43)

The miss penalty on the A53 is 13 clock cycles for the L1 cache and 124 for the L2 cache



Characteristic	ARM Cortex-A53	Intel Core i7
L1 cache organization	Split instruction and data caches	Split instruction and data caches
L1 cache size	8-64 KiB each for instructions/data	32 KiB each for instructions/data per core
L1 cache associativity	2-way (I), 2-way (D) set associative	8-way (I), 8-way (D) set associative
L1 replacement	Random	Approximated LRU
L1 block size	64 bytes	64 bytes
L1 write policy	Write-back, Write-allocate(?)	Write-back, No-write-allocate
L1 hit time (load-use)	1 clock cycle	4 clock cycles, pipelined
L2 cache organization	Unified (instruction and data)	Unified (instruction and data) per core
L2 cache size	128 KiB to 2 MiB	256 KiB (0.25 MiB)
L2 cache associativity	8-way set associative	4-way set associative
L2 replacement	Approximated LRU	Approximated LRU
L2 block size	64 bytes	64 bytes
L2 write policy	Write-back, Write-allocate	Write-back, Write-allocate
L2 hit time	11 clock cycles	12 clock cycles
L3 cache organization	–	Unified (instruction and data)
L3 cache size	–	2 MiB/core, shared
L3 cache associativity	–	16-way set associative
L3 replacement	–	Approximated LRU
L3 block size	–	64 bytes
L3 write policy	–	Write-back, Write-allocate
L3 hit time	–	44 clock cycles

CSUN
CALIFORNIA
STATE UNIVERSITY
NORTHRIDGE

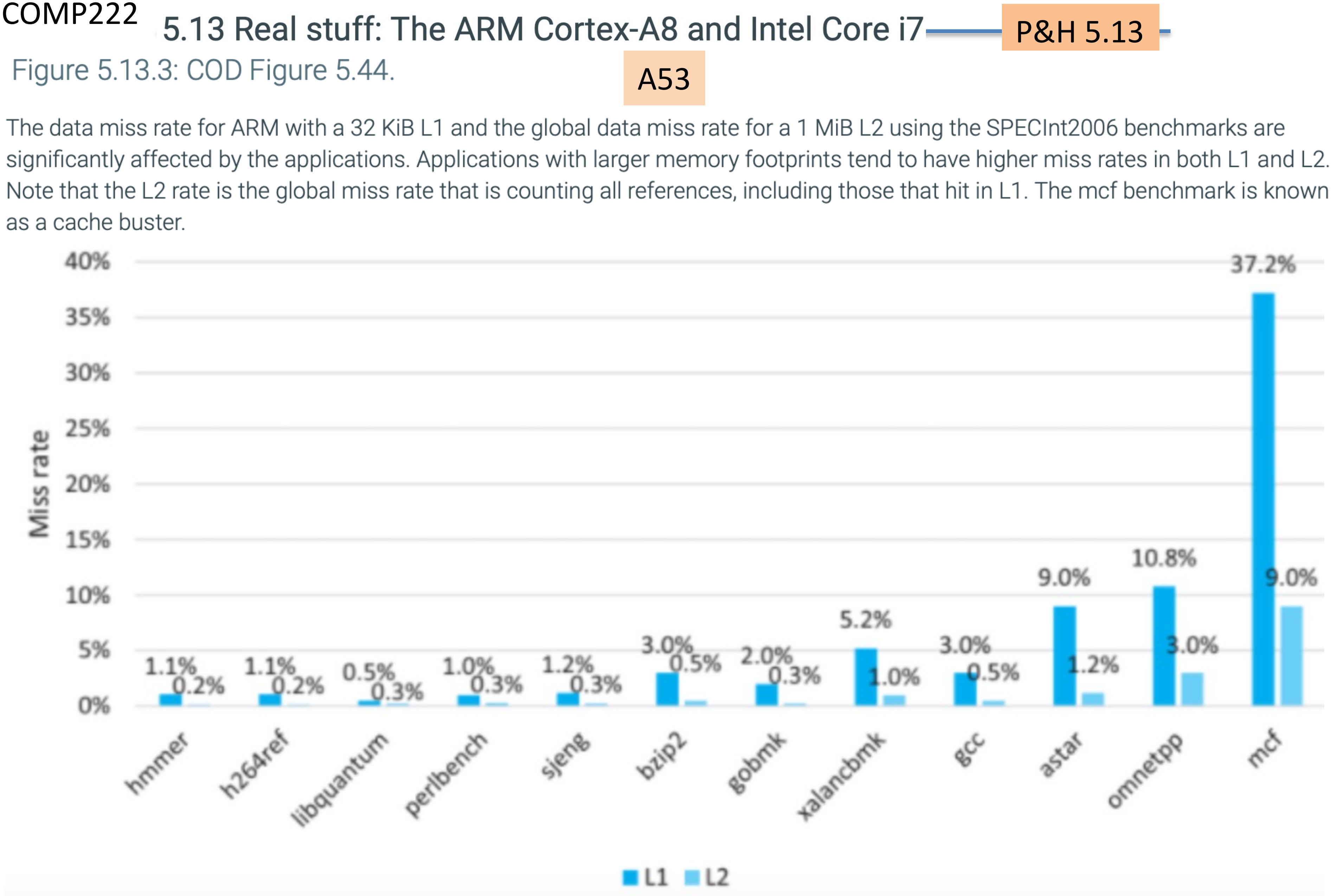
COMP222

Cache Data

DR JEFF SOFTWARE
INDIE APP DEVELOPER
© Jeff Drobman
2020-23

P&H 5.13

A53



5.13 Real stuff: The ARM Cortex-A8 and Intel Core i7

Figure 5.13.3: COD Figure 5.44.

The data miss rate for ARM with a 32 KiB L1 and the global data miss rate for a 1 MiB L2 using the SPECInt2006 benchmarks are significantly affected by the applications. Applications with larger memory footprints tend to have higher miss rates in both L1 and L2. Note that the L2 rate is the global miss rate that is counting all references, including those that hit in L1. The mcf benchmark is known as a cache buster.

Cache Data

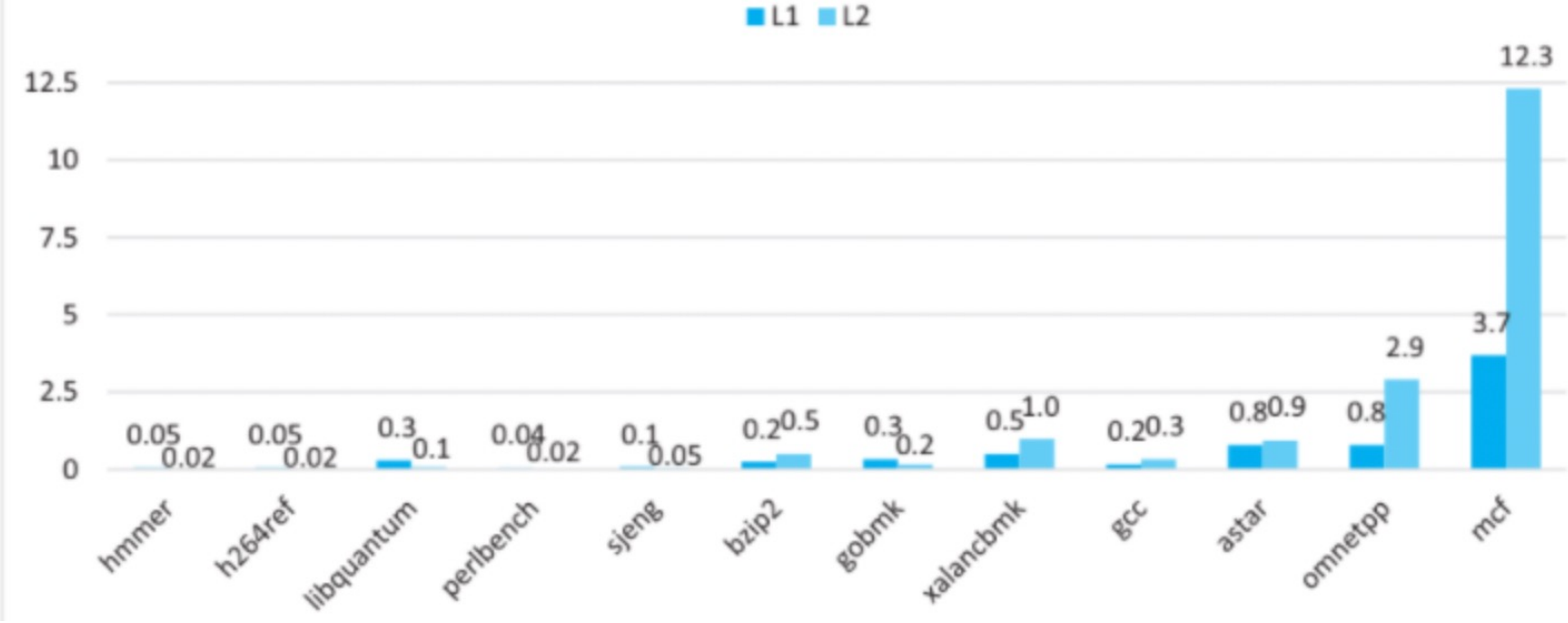
5.13 Real stuff: The ARM Cortex-A8 and Intel Core i7

P&H 5.13

Figure 5.13.4: COD Figure 5.45.

A53

The average memory access penalty per data memory reference coming from L1 and L2 is shown for the A53 processor when running SPECInt2006. Although the miss rates for L1 are significantly higher, the L2 miss penalty, which is more than five times higher, means that the L2 misses can contribute significantly.



Cache Data

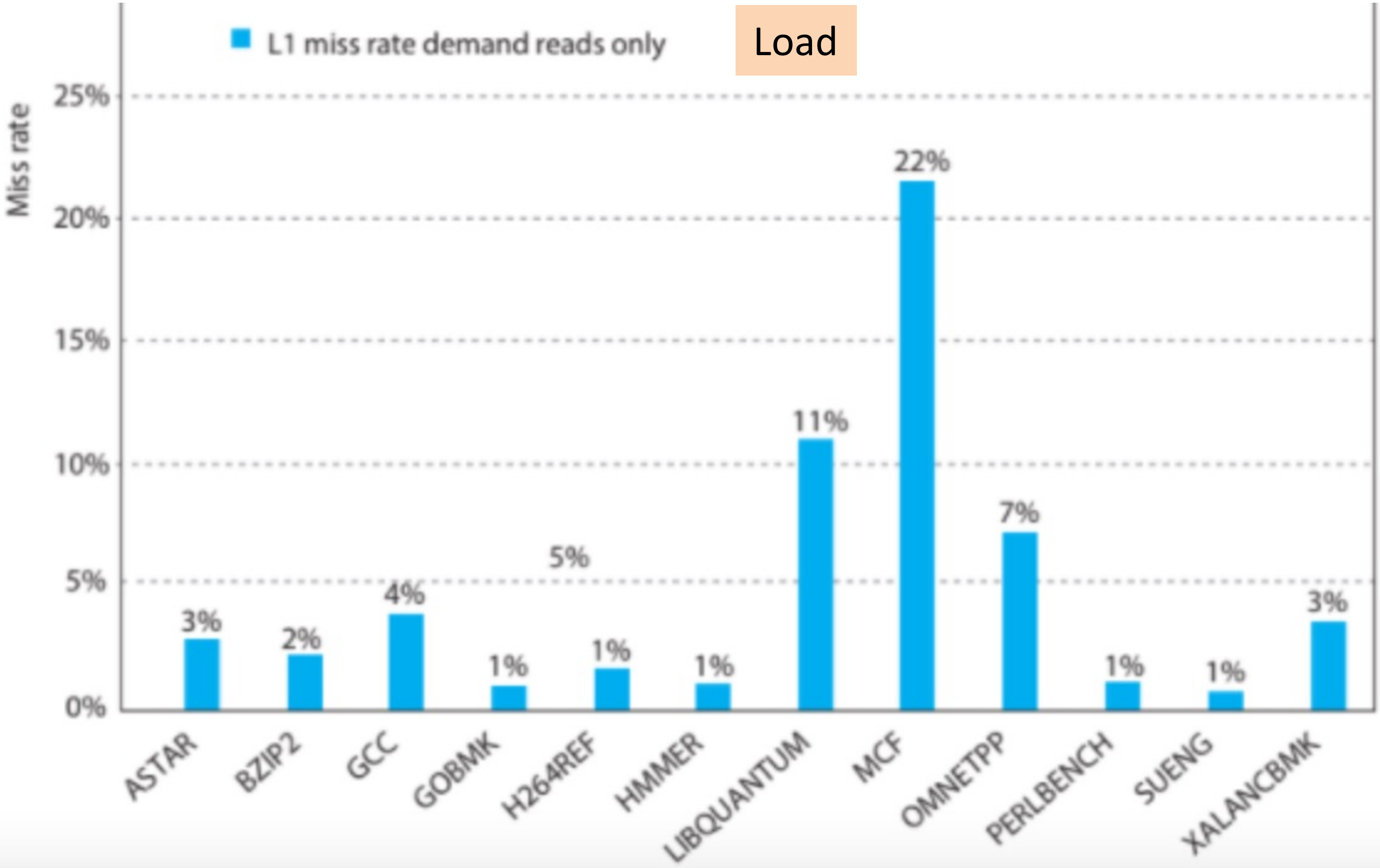
5.13 Real stuff: The ARM Cortex-A8 and Intel Core i7

P&H 5.13

Figure 5.13.5 memory hierarchies

A53

The L1 data cache miss rate for the SPECint2006 benchmarks is shown in two ways relative to the demand L1 reads for demand accesses (excluding prefetched). These data, like the rest in this section, were collected by Professor Lu Peng and PhD student Qun Liu, both of Louisiana State University (see Peng et al., 2008)



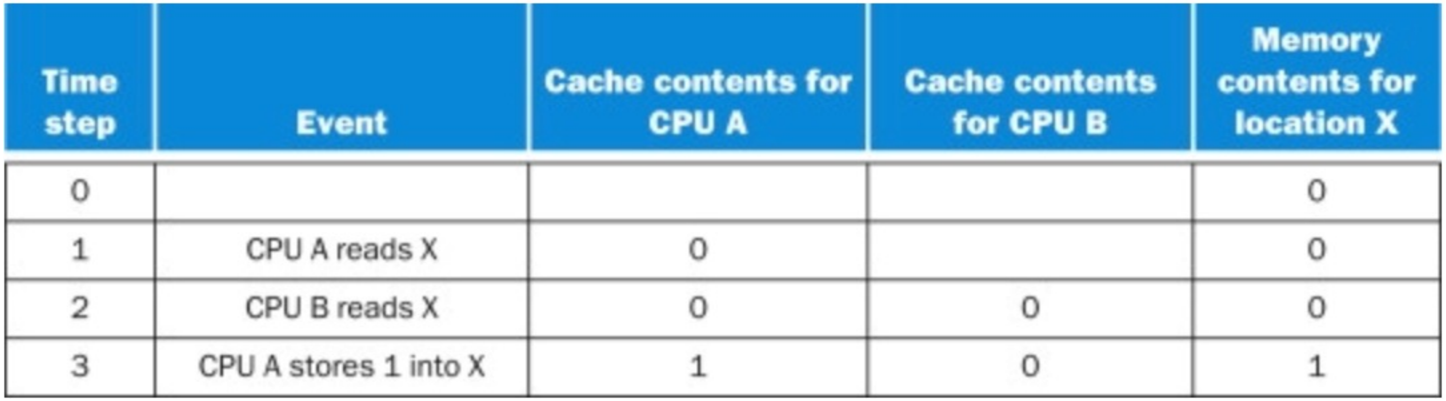
Load

Cache Coherency

5.10 Parallelism and memory hierarchies: Cache coherence

P&H 5.10

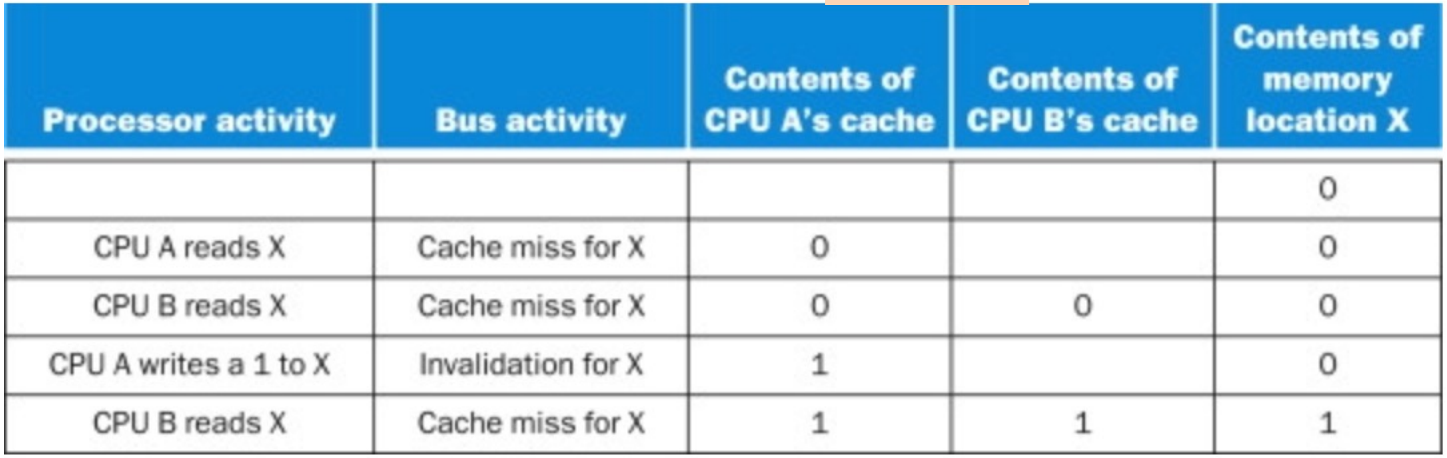
Figure 5.10.1: The cache coherence problem for a single memory location (X), read and written by two processors (A and B) (COD Figure 5.40).



Time step	Event	Cache contents for CPU A	Cache contents for CPU B	Memory contents for location X
0				0
1	CPU A reads X	0		0
2	CPU B reads X	0	0	0
3	CPU A stores 1 into X	1	0	1

Figure 5.10.2: An example of an invalidation protocol working on a snooping bus for a single cache block (X) with write-back caches (COD Figure 5.41).

Valid bit



Processor activity	Bus activity	Contents of CPU A's cache	Contents of CPU B's cache	Contents of memory location X
				0
CPU A reads X	Cache miss for X	0		0
CPU B reads X	Cache miss for X	0	0	0
CPU A writes a 1 to X	Invalidation for X	1		0
CPU B reads X	Cache miss for X	1	1	1

Computer Architecture

Interrupts

(see separate slide set ***122 Lecture 2***)

Interrupts

General MIPS

CLASSES

- ❖ **MASKABLE**
  - ❑ NMI (non-maskable)
    - ▪ Power-ON Reset
  - ❑ INT (maskable)
- ❖ **VECTORED**
  - ❑ NVI (non-V)
  - ❑ VI
- ❖ PRIORITY (PIC)
  - ❑ High
  - ❑ Low (High INTs "preempt" Low)
- ❖ INTERNAL
  - ❑ Hardware events
    - ▪ Timers
    - ▪ ADC
    - ▪ I/O (S, P)
  - ❑ Software exceptions

INT's (8)

- ◆ INT 0 (Pin 33)
- ◆ INT 1 (Pin 34)
- ◆ INT 2 (Pin 35)
- ◆ INT 7

ENABLES

- ❖ GIE (2) – global (2 groups)
- ❖ *Mask*: INT 0-7

PRIORITIES

- ❖ HIGH
- ❖ LOW

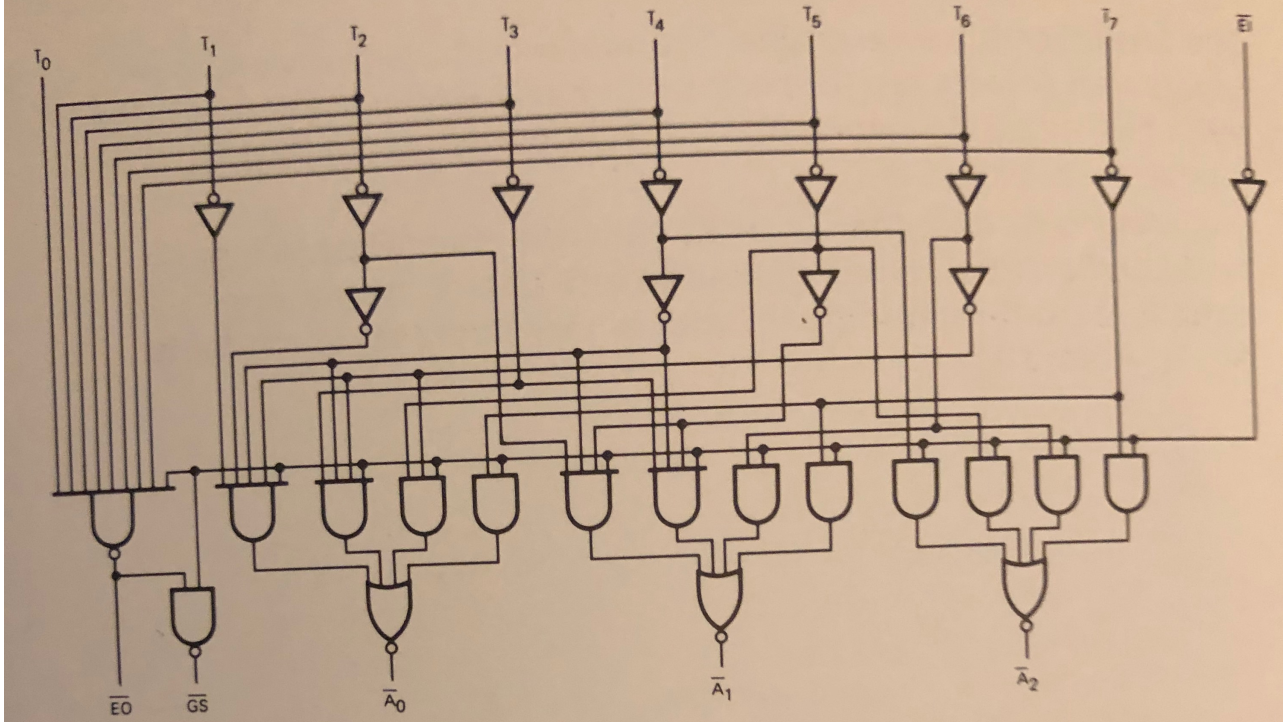
→ SAVED ON STACK

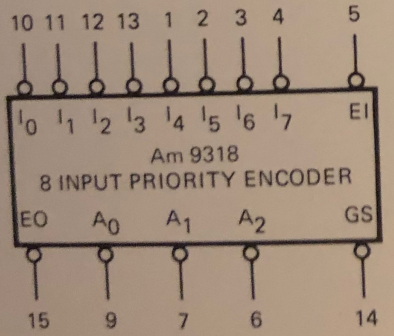
PROCESSOR STATE

- ❖ **PC**
- ❖ STATUS
- ❖ CAUSE

PIC: Priority Interrupt Enc

Logic Diagram/Symbol





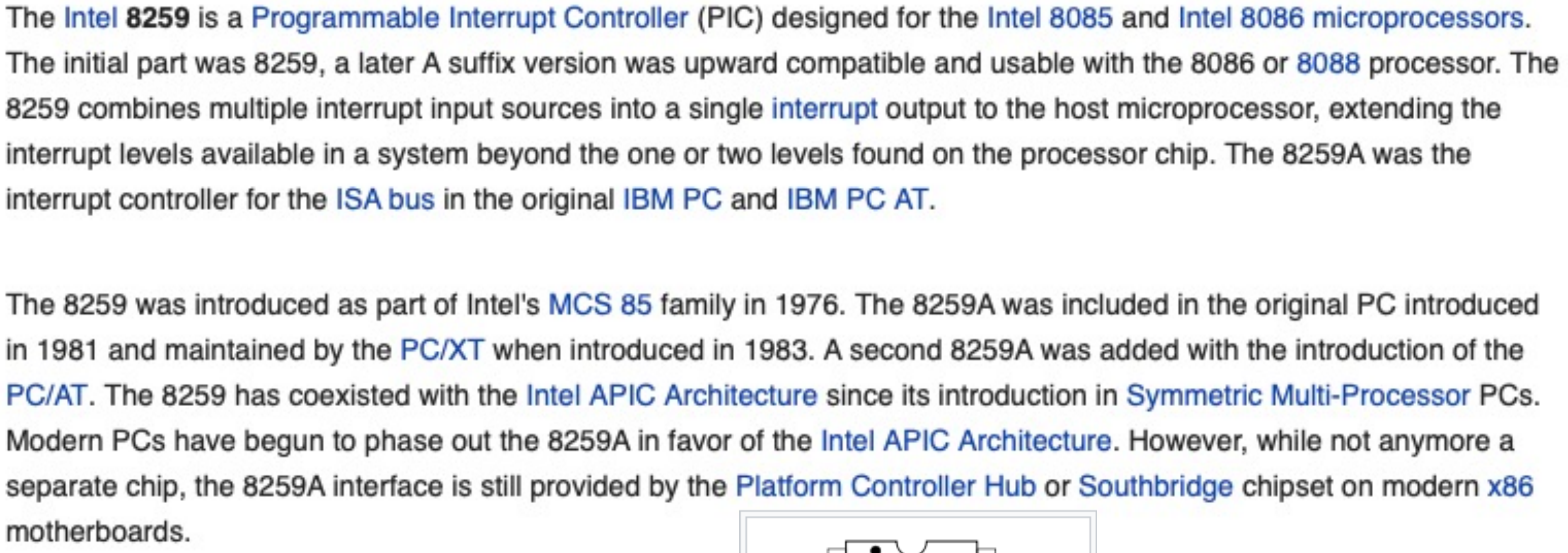
Characteristics

Typical Delay $\overline{I}$ to $\overline{A}$ 16 ns

Typical Power Dissipation 250 mW

V_{CC} = Pin 16
GND = Pin 8

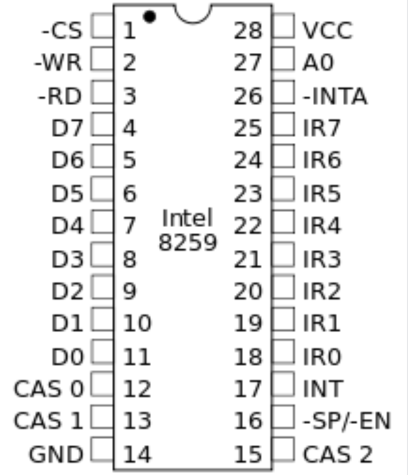
I8259 PIC



The Intel **8259** is a Programmable Interrupt Controller (PIC) designed for the Intel 8085 and Intel 8086 microprocessors. The initial part was 8259, a later A suffix version was upward compatible and usable with the 8086 or 8088 processor. The 8259 combines multiple interrupt input sources into a single interrupt output to the host microprocessor, extending the interrupt levels available in a system beyond the one or two levels found on the processor chip. The 8259A was the interrupt controller for the ISA bus in the original IBM PC and IBM PC AT.

The 8259 was introduced as part of Intel's MCS 85 family in 1976. The 8259A was included in the original PC introduced in 1981 and maintained by the PC/XT when introduced in 1983. A second 8259A was added with the introduction of the PC/AT. The 8259 has coexisted with the Intel APIC Architecture since its introduction in Symmetric Multi-Processor PCs. Modern PCs have begun to phase out the 8259A in favor of the Intel APIC Architecture. However, while not anymore a separate chip, the 8259A interface is still provided by the Platform Controller Hub or Southbridge chipset on modern x86 motherboards.

Closeup of an Intel 8259A IRQ chip from a PC XT.



Pinout

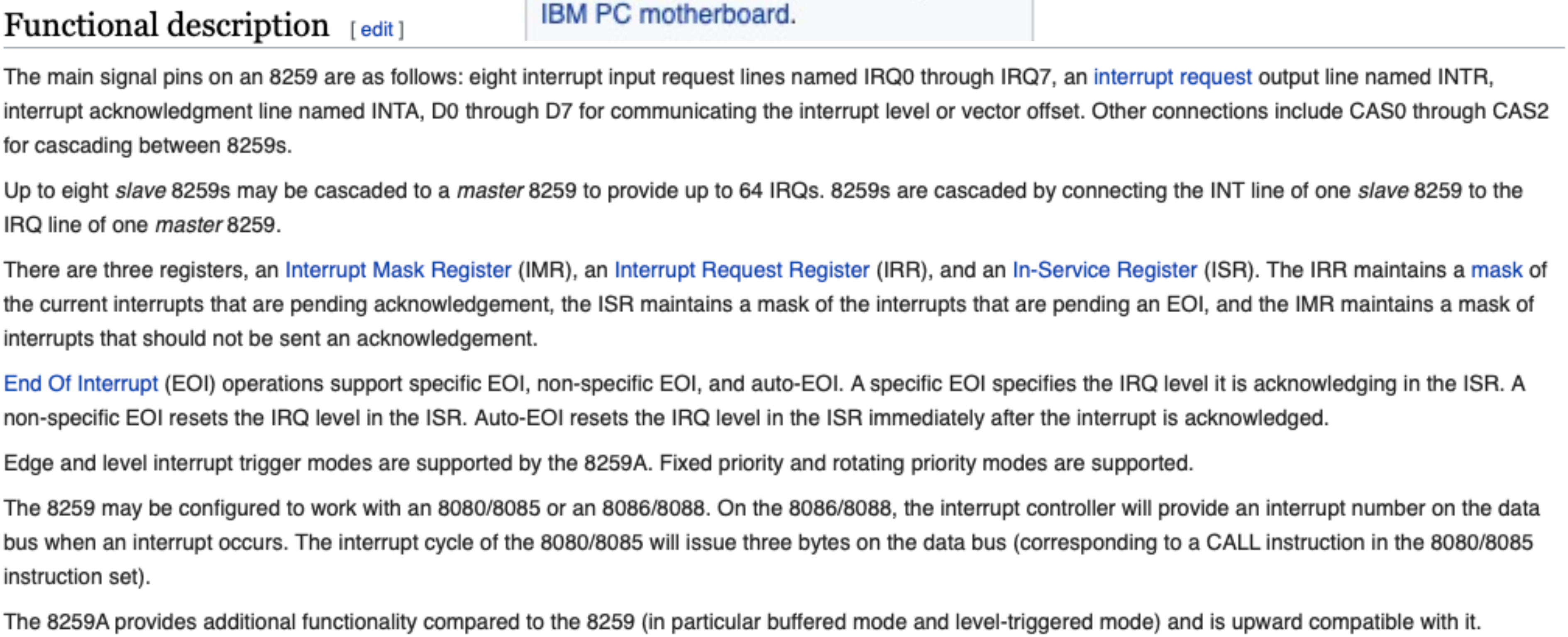
- IR0-7
- D0-7
- CAS0-3
- INTReq

I8259 PIC

- ➢ Trigger: Edge vs. Level
- ➢ Priority: Fixed vs. Rotating

NEC D8259AC, used on the original IBM PC motherboard.

- ❖ IR0-7
- ❖ D0-7
- ❖ CAS0-3
- ❖ INTReq



Functional description [edit]

The main signal pins on an 8259 are as follows: eight interrupt input request lines named IRQ0 through IRQ7, an interrupt request output line named INTR, interrupt acknowledgment line named INTA, D0 through D7 for communicating the interrupt level or vector offset. Other connections include CAS0 through CAS2 for cascading between 8259s.

Up to eight *slave* 8259s may be cascaded to a *master* 8259 to provide up to 64 IRQs. 8259s are cascaded by connecting the INT line of one *slave* 8259 to the IRQ line of one *master* 8259.

There are three registers, an Interrupt Mask Register (IMR), an Interrupt Request Register (IRR), and an In-Service Register (ISR). The IRR maintains a mask of the current interrupts that are pending acknowledgement, the ISR maintains a mask of the interrupts that are pending an EOI, and the IMR maintains a mask of interrupts that should not be sent an acknowledgement.

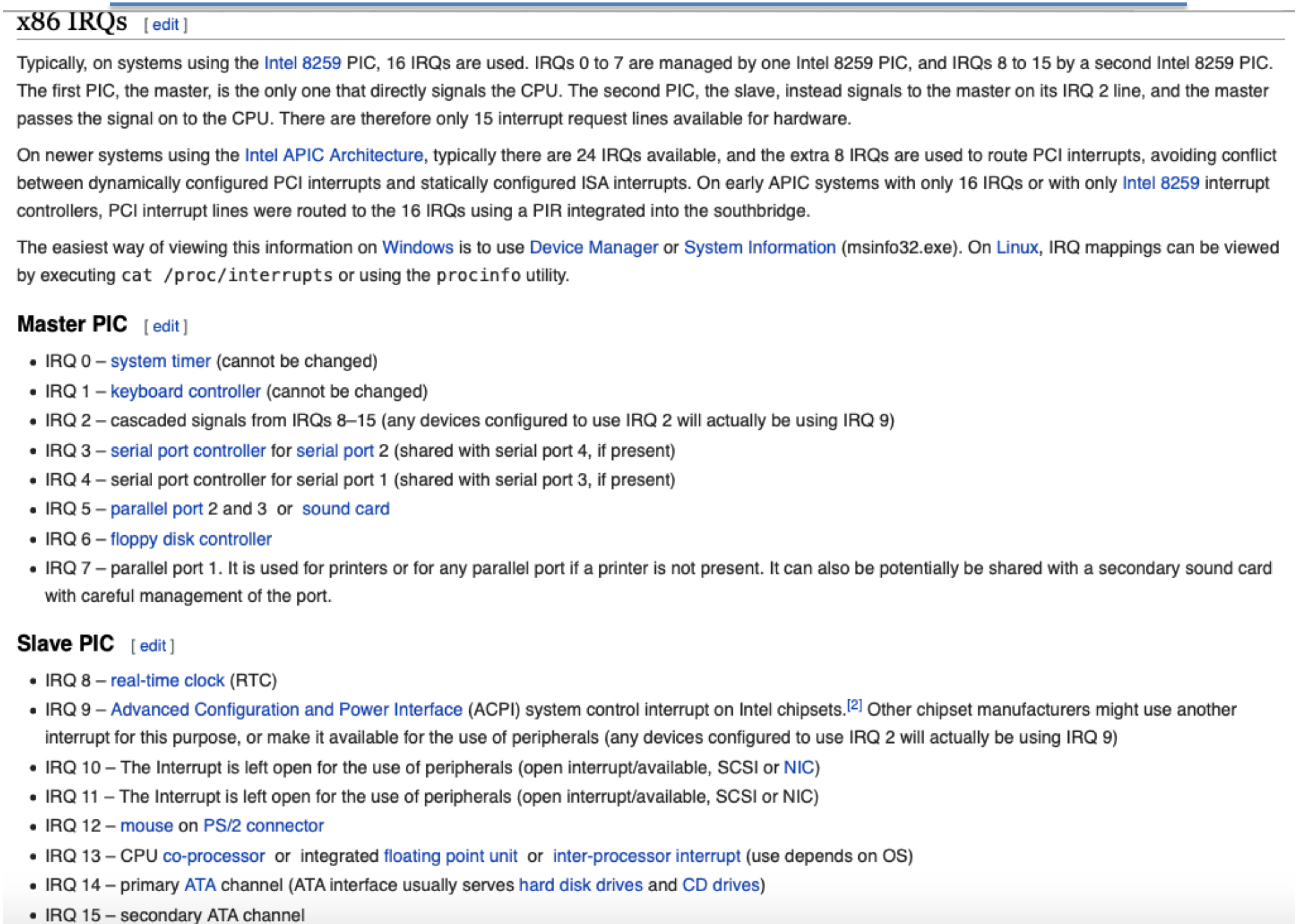
End Of Interrupt (EOI) operations support specific EOI, non-specific EOI, and auto-EOI. A specific EOI specifies the IRQ level it is acknowledging in the ISR. A non-specific EOI resets the IRQ level in the ISR. Auto-EOI resets the IRQ level in the ISR immediately after the interrupt is acknowledged.

Edge and level interrupt trigger modes are supported by the 8259A. Fixed priority and rotating priority modes are supported.

The 8259 may be configured to work with an 8080/8085 or an 8086/8088. On the 8086/8088, the interrupt controller will provide an interrupt number on the data bus when an interrupt occurs. The interrupt cycle of the 8080/8085 will issue three bytes on the data bus (corresponding to a CALL instruction in the 8080/8085 instruction set).

The 8259A provides additional functionality compared to the 8259 (in particular buffered mode and level-triggered mode) and is upward compatible with it.

i8259 PIC



x86 IRQs [edit]

Typically, on systems using the Intel 8259 PIC, 16 IRQs are used. IRQs 0 to 7 are managed by one Intel 8259 PIC, and IRQs 8 to 15 by a second Intel 8259 PIC. The first PIC, the master, is the only one that directly signals the CPU. The second PIC, the slave, instead signals to the master on its IRQ 2 line, and the master passes the signal on to the CPU. There are therefore only 15 interrupt request lines available for hardware.

On newer systems using the Intel APIC Architecture, typically there are 24 IRQs available, and the extra 8 IRQs are used to route PCI interrupts, avoiding conflict between dynamically configured PCI interrupts and statically configured ISA interrupts. On early APIC systems with only 16 IRQs or with only Intel 8259 interrupt controllers, PCI interrupt lines were routed to the 16 IRQs using a PIR integrated into the southbridge.

The easiest way of viewing this information on Windows is to use Device Manager or System Information (msinfo32.exe). On Linux, IRQ mappings can be viewed by executing `cat /proc/interrupts` or using the `procinfo` utility.

Master PIC [edit]

- IRQ 0 – system timer (cannot be changed)
- IRQ 1 – keyboard controller (cannot be changed)
- IRQ 2 – cascaded signals from IRQs 8–15 (any devices configured to use IRQ 2 will actually be using IRQ 9)
- IRQ 3 – serial port controller for serial port 2 (shared with serial port 4, if present)
- IRQ 4 – serial port controller for serial port 1 (shared with serial port 3, if present)
- IRQ 5 – parallel port 2 and 3 or sound card
- IRQ 6 – floppy disk controller
- IRQ 7 – parallel port 1. It is used for printers or for any parallel port if a printer is not present. It can also be potentially be shared with a secondary sound card with careful management of the port.

Slave PIC [edit]

- IRQ 8 – real-time clock (RTC)
- IRQ 9 – Advanced Configuration and Power Interface (ACPI) system control interrupt on Intel chipsets.[2] Other chipset manufacturers might use another interrupt for this purpose, or make it available for the use of peripherals (any devices configured to use IRQ 2 will actually be using IRQ 9)
- IRQ 10 – The Interrupt is left open for the use of peripherals (open interrupt/available, SCSI or NIC)
- IRQ 11 – The Interrupt is left open for the use of peripherals (open interrupt/available, SCSI or NIC)
- IRQ 12 – mouse on PS/2 connector
- IRQ 13 – CPU co-processor or integrated floating point unit or inter-processor interrupt (use depends on OS)
- IRQ 14 – primary ATA channel (ATA interface usually serves hard disk drives and CD drives)
- IRQ 15 – secondary ATA channel

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP222

Lab 4 INT Model

DR JEFF SOFTWARE
INDIE APP DEVELOPER
© Jeff Drobman
2020-23

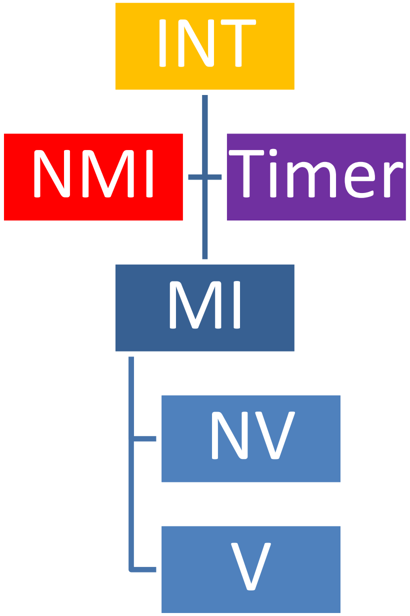
INT's Used: 4

- **MASKABLE (3)**
  - ❑ **1** NMI (non-maskable)
    - Power-ON Reset
  - ❑ **2** INT (maskable)

- **VECTORED (2)**
  - ❑ **1** NVI (non)
  - ❑ **1** VI

- **TIMER (1)**

Hierarchy: ***Priority***



Interrupt Handlers

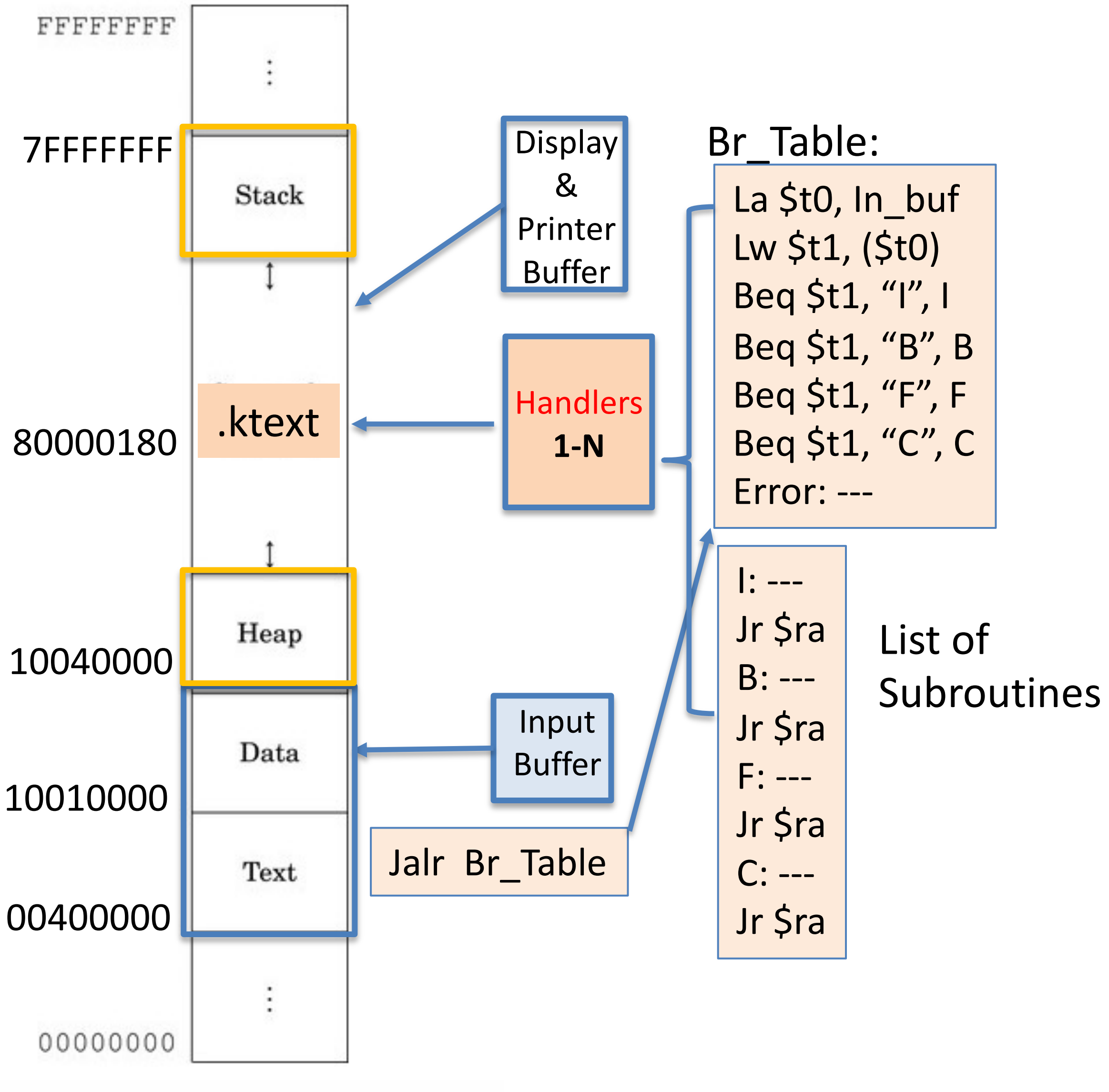
Lab 4

- Decode *Pending* Interrupts
- Allocate memory for Handlers
- Use *Jump Table*
  - ❑ Order by *Priority*
  - ❑ Test & Jump
  - ❑ Handlers as subroutines: jal → jr $ra

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP222

Lab 4: Memory

Lab 4

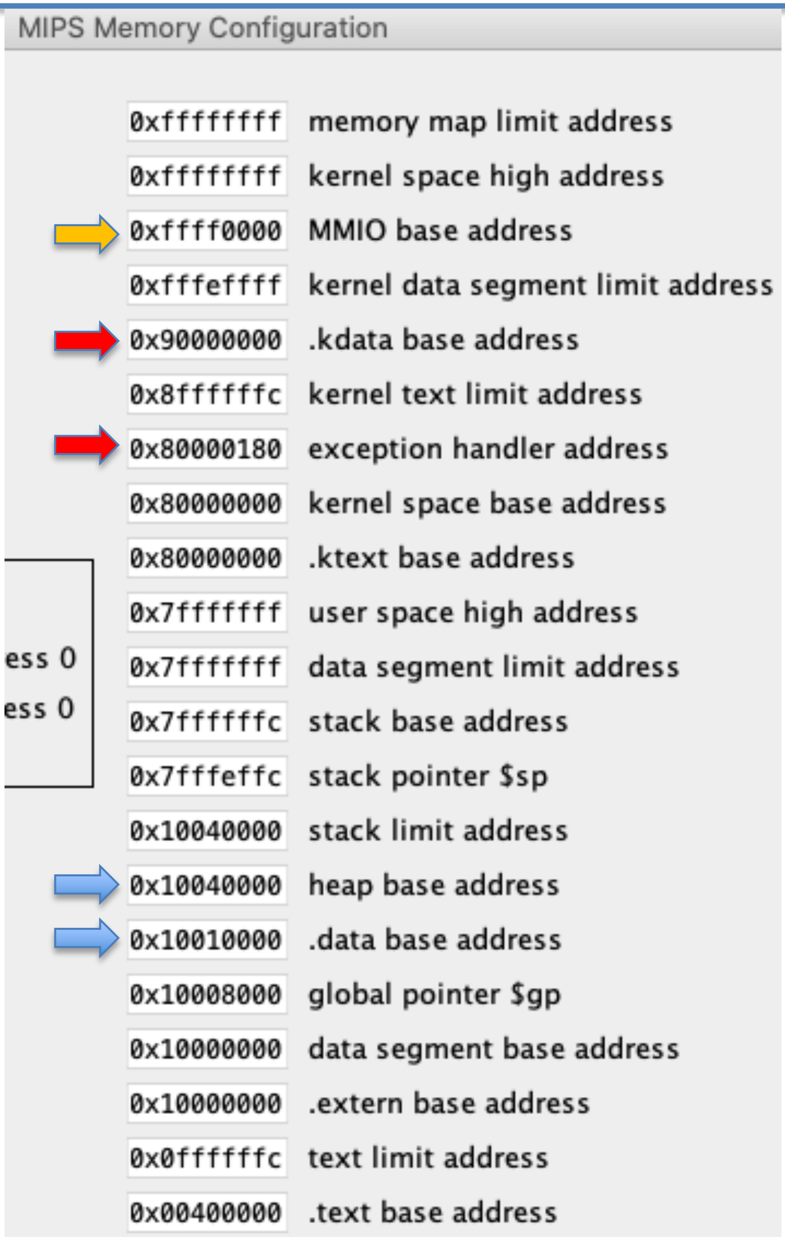


```
La $t0, In_buf
Lw $t1, ($t0)
Beq $t1, “I”, I
Beq $t1, “B”, B
Beq $t1, “F”, F
Beq $t1, “C”, C
Error: ---
```

```
I: ---
Jr $ra
B: ---
Jr $ra
F: ---
Jr $ra
C: ---
Jr $ra
```

Typical memory layout for a program with a 32-bit address space.

MIPS Memory Config



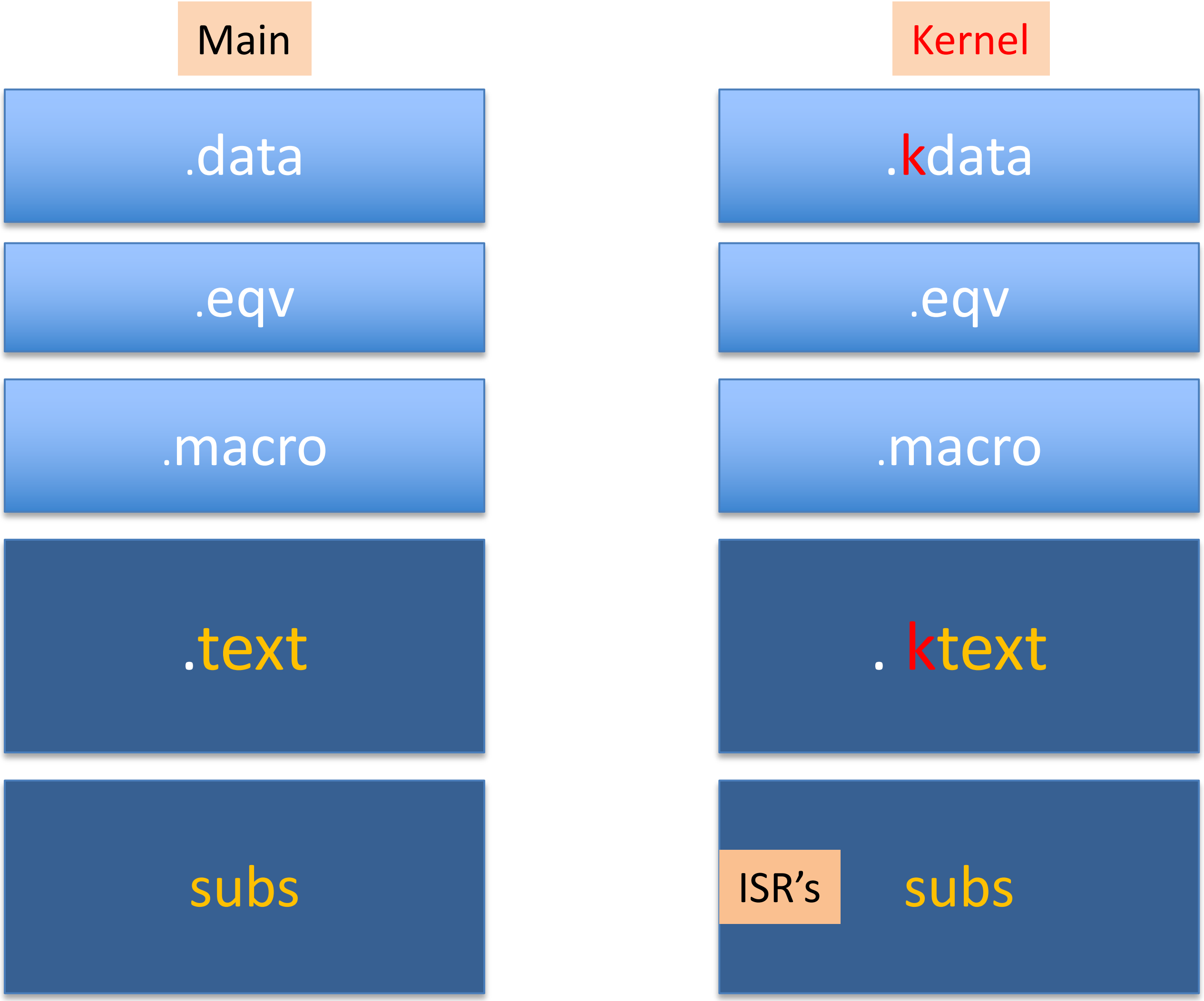
MIPS Memory Configuration

ess 0
ess 0

0xffffffff	memory map limit address
0xffffffff	kernel space high address
0xffff0000	MMIO base address
0xfffeffff	kernel data segment limit address
0x90000000	.kdata base address
0x8ffffffc	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
0x7ffffffc	stack base address
0x7fffeffc	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
0x0ffffffc	text limit address
0x00400000	.text base address

Lab 4 Program Structure

Lab 4

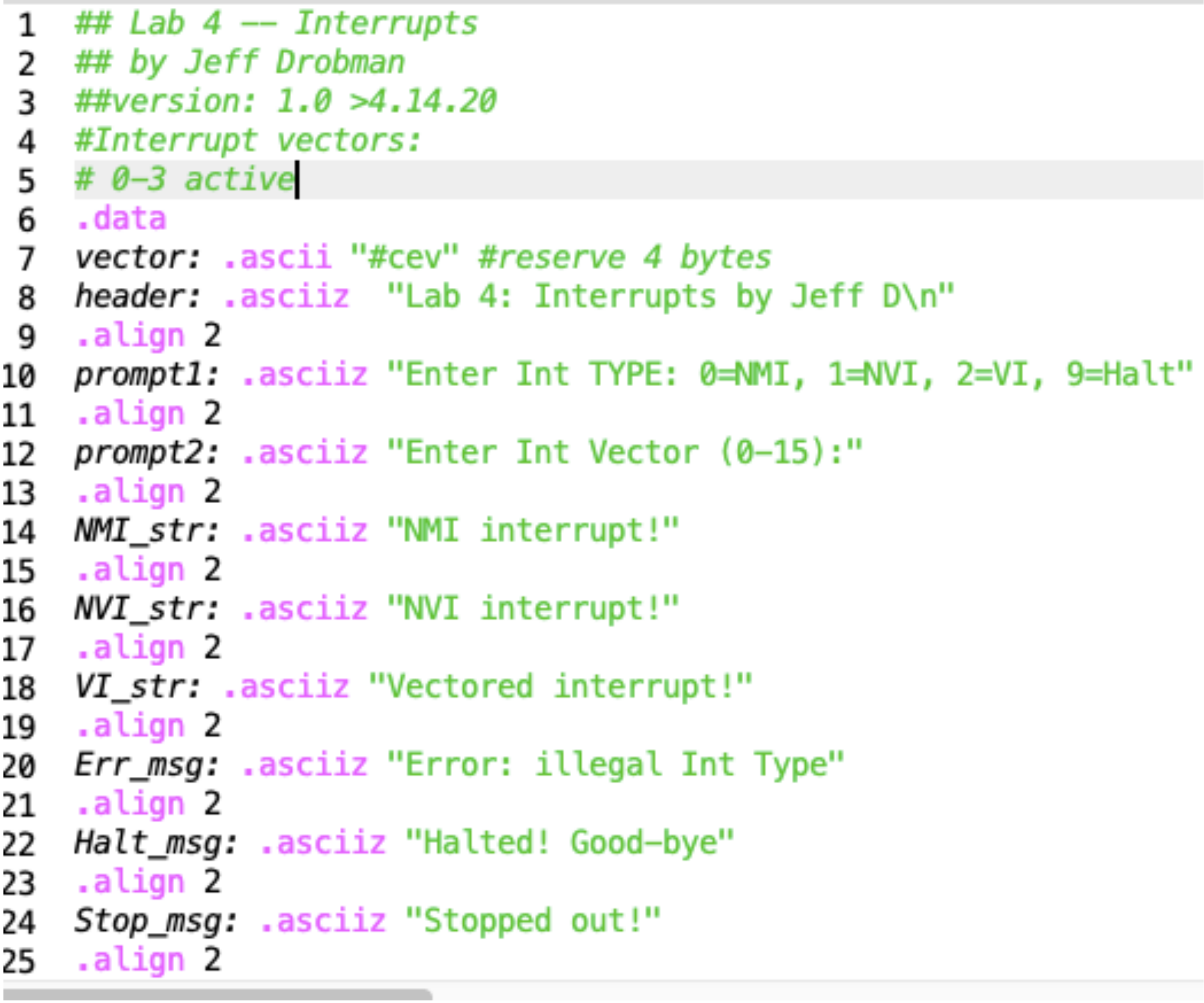


Main
- .data
- .eqv
- .macro
- .text
- subs

Kernel
- .kdata
- .eqv
- .macro
- . ktext
- ISR's subs

Lab 4 Program

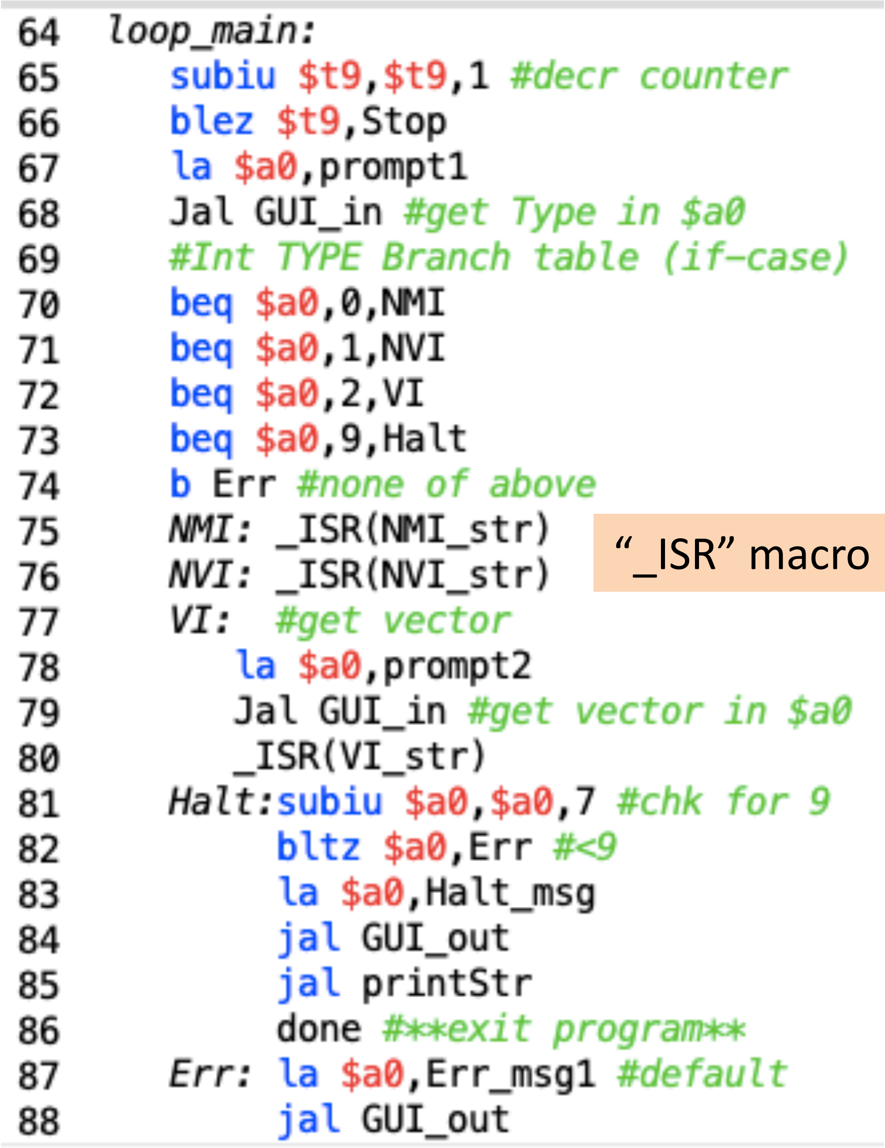
Lab 4



```
## Lab 4 -- Interrupts
## by Jeff Drobman
##version: 1.0 >4.14.20
#Interrupt vectors:
# 0-3 active
.data
vector: .ascii "#cev" #reserve 4 bytes
header: .asciiz  "Lab 4: Interrupts by Jeff D\n"
.align 2
prompt1: .asciiz "Enter Int TYPE: 0=NMI, 1=NVI, 2=VI, 9=Halt"
.align 2
prompt2: .asciiz "Enter Int Vector (0-15):"
.align 2
NMI_str: .asciiz "NMI interrupt!"
.align 2
NVI_str: .asciiz "NVI interrupt!"
.align 2
VI_str: .asciiz "Vectored interrupt!"
.align 2
Err_msg: .asciiz "Error: illegal Int Type"
.align 2
Halt_msg: .asciiz "Halted! Good-bye"
.align 2
Stop_msg: .asciiz "Stopped out!"
.align 2
```

Lab 4 Code Loop

Lab 4

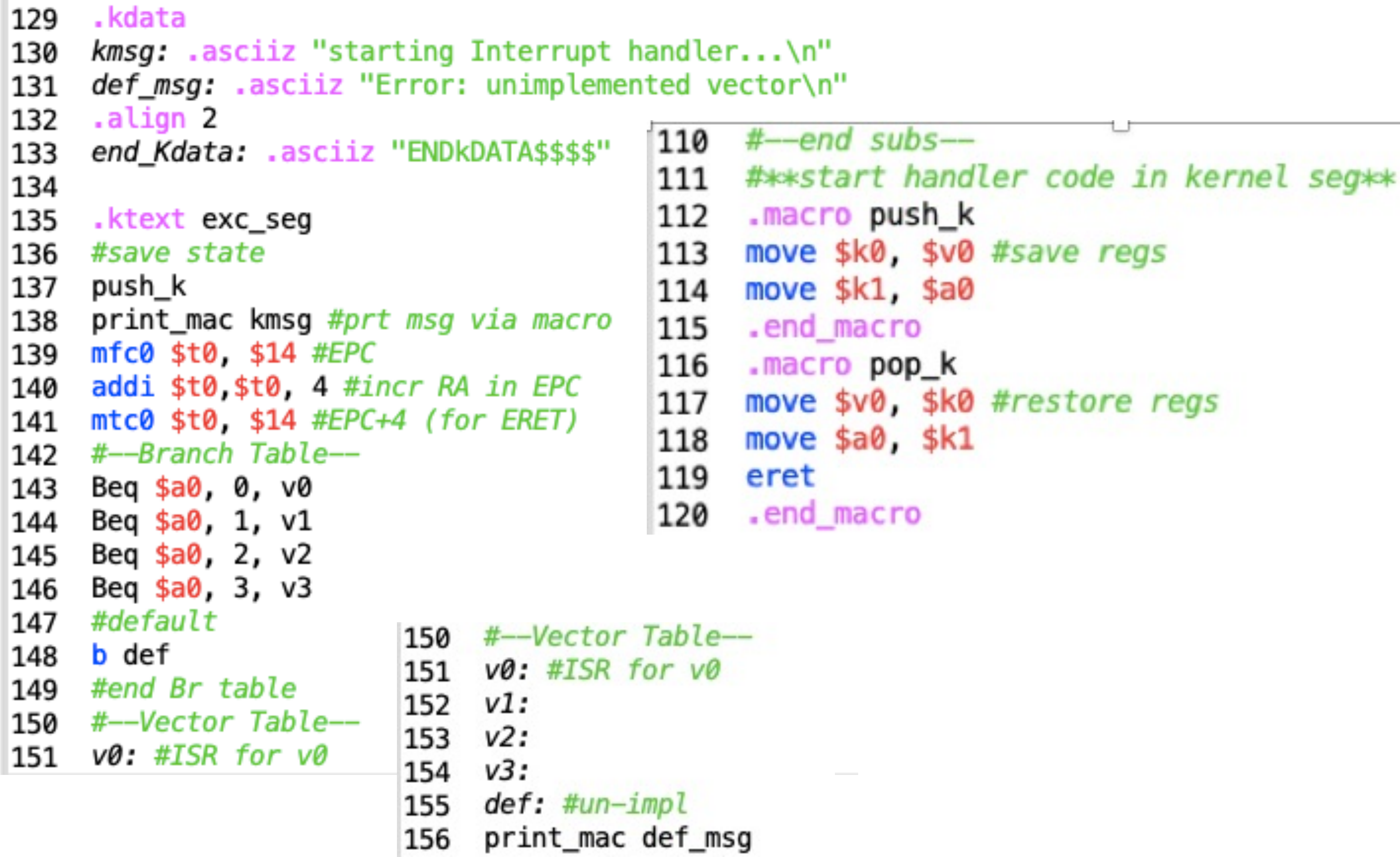


```
loop_main:
   subiu $t9,$t9,1 #decr counter
   blez $t9,Stop
   la $a0,prompt1
   Jal GUI_in #get Type in $a0
   #Int TYPE Branch table (if-case)
   beq $a0,0,NMI
   beq $a0,1,NVI
   beq $a0,2,VI
   beq $a0,9,Halt
   b Err #none of above
   NMI: _ISR(NMI_str)
   NVI: _ISR(NVI_str)
   VI:  #get vector
      la $a0,prompt2
      Jal GUI_in #get vector in $a0
      _ISR(VI_str)
   Halt:subiu $a0,$a0,7 #chk for 9
        bltz $a0,Err #<9
        la $a0,Halt_msg
        jal GUI_out
        jal printStr
        done #**exit program**
   Err: la $a0,Err_msg1 #default
        jal GUI_out
```

"_ISR" macro

Lab 4 Kernel Code/Data

Lab 4



```
.kdata
kmsg: .asciiz "starting Interrupt handler...\n"
def_msg: .asciiz "Error: unimplemented vector\n"
.align 2
end_Kdata: .asciiz "ENDkDATA$$$$"

.ktext exc_seg
#save state
push_k
print_mac kmsg #prt msg via macro
mfc0 $t0, $14 #EPC
addi $t0,$t0, 4 #incr RA in EPC
mtc0 $t0, $14 #EPC+4 (for ERET)
#--Branch Table--
Beq $a0, 0, v0
Beq $a0, 1, v1
Beq $a0, 2, v2
Beq $a0, 3, v3
#default
b def
#end Br table
#--Vector Table--
v0: #ISR for v0
```

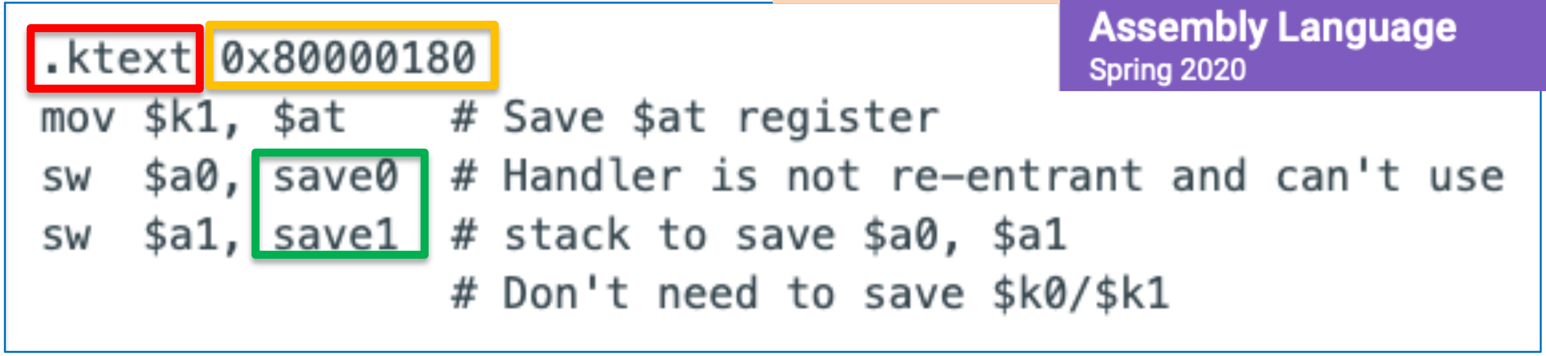
```
#--end subs--
#**start handler code in kernel seg**
.macro push_k
move $k0, $v0 #save regs
move $k1, $a0
.end_macro
.macro pop_k
move $v0, $k0 #restore regs
move $a0, $k1
eret
.end_macro
```

```
#--Vector Table--
v0: #ISR for v0
v1:
v2:
v3:
def: #un-impl
print_mac def_msg
```

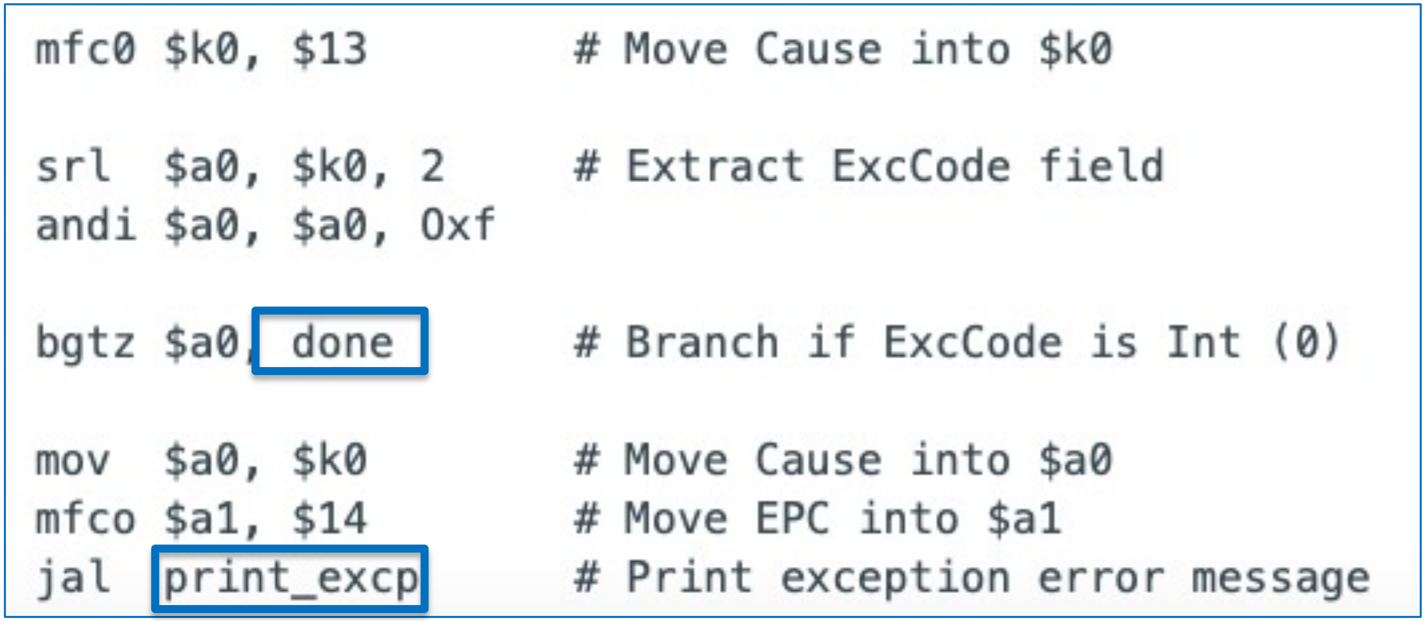
Interrupts & Exceptions

P&H Ch 7

COMP 122: Computer Architecture and Assembly Language
Spring 2020



```
.ktext 0x80000180
mov $k1, $at      # Save $at register
sw  $a0, save0    # Handler is not re-entrant and can't use
sw  $a1, save1    # stack to save $a0, $a1
                  # Don't need to save $k0/$k1
```

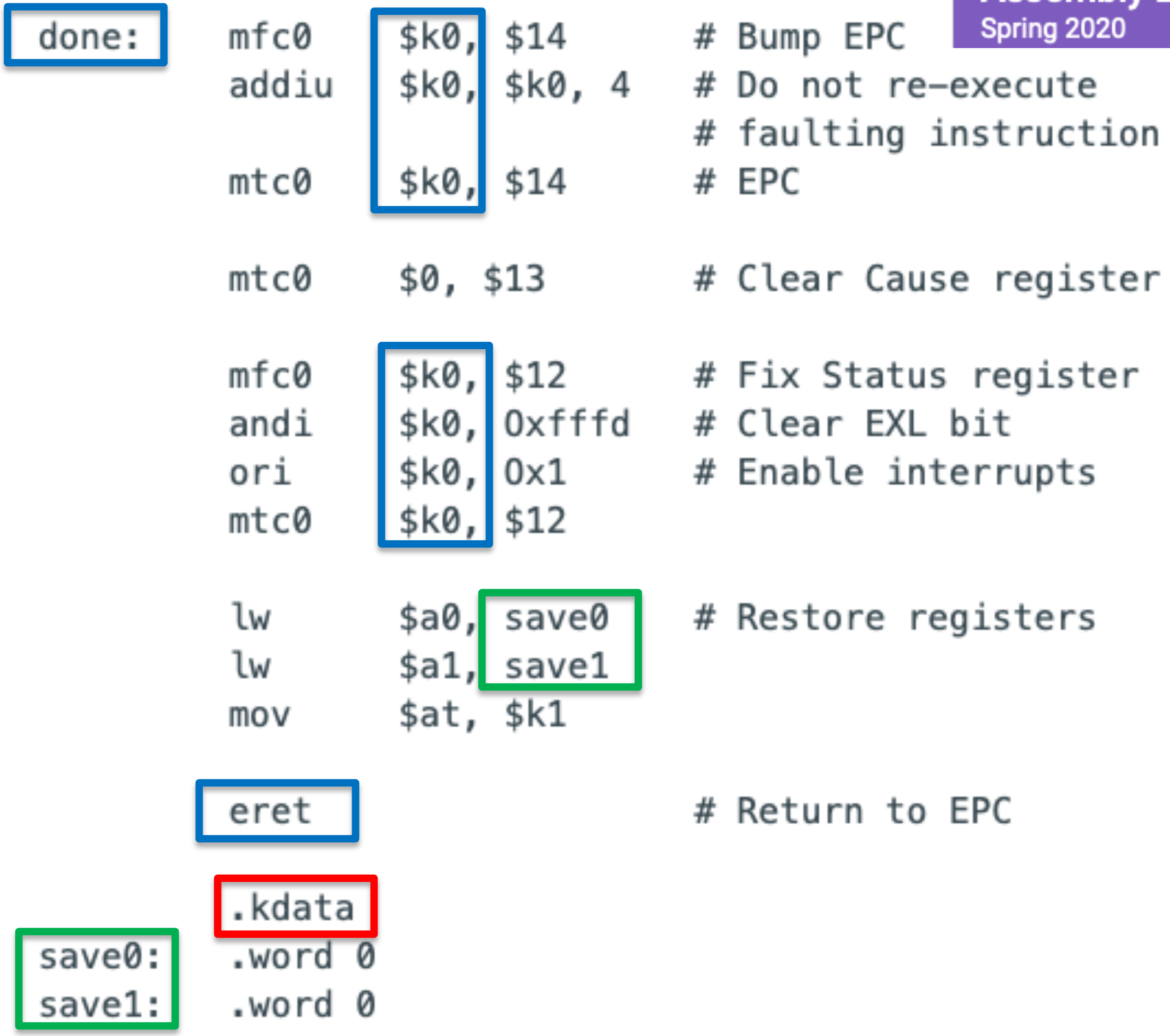


```
mfc0 $k0, $13          # Move Cause into $k0

srl  $a0, $k0, 2       # Extract ExcCode field
andi $a0, $a0, 0xf

bgtz $a0, done         # Branch if ExcCode is Int (0)

mov  $a0, $k0          # Move Cause into $a0
mfco $a1, $14          # Move EPC into $a1
jal  print_excp        # Print exception error message
```

Interrupts & Exceptions

P&H Ch 7

COMP 122: Computer Architecture and Assembly Language
Spring 2020



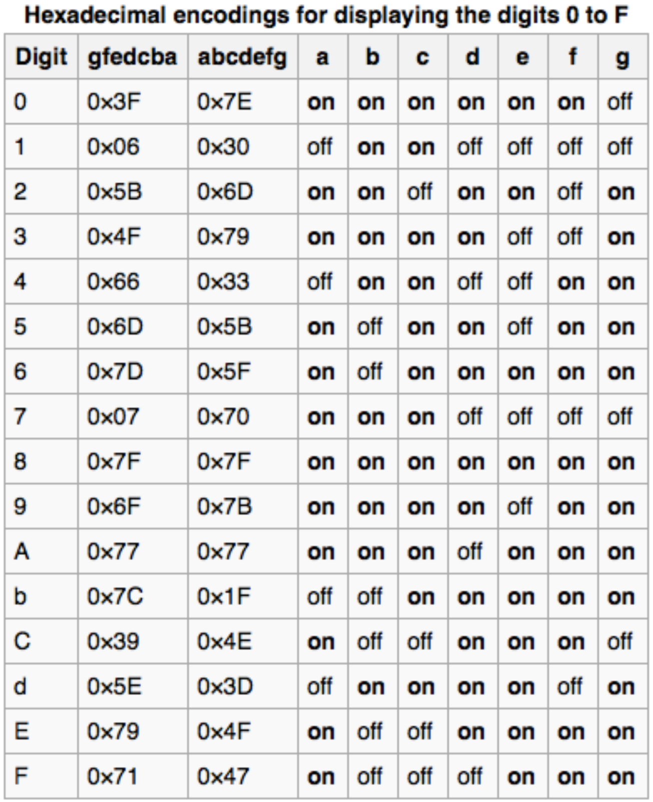
```
done:   mfc0    $k0, $14        # Bump EPC
        addiu   $k0, $k0, 4     # Do not re-execute
                                # faulting instruction
        mtc0    $k0, $14        # EPC

        mtc0    $0, $13         # Clear Cause register

        mfc0    $k0, $12        # Fix Status register
        andi    $k0, 0xfffd     # Clear EXL bit
        ori     $k0, 0x1        # Enable interrupts
        mtc0    $k0, $12

        lw      $a0, save0      # Restore registers
        lw      $a1, save1
        mov     $at, $k1

        eret                    # Return to EPC

        .kdata
save0:  .word 0
save1:  .word 0
```

Project

APPLICATION – **LCD/LED Modules**

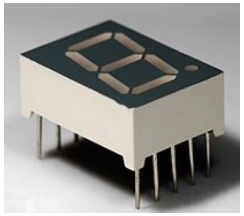
send letters/numbers to a 7-segment LCD

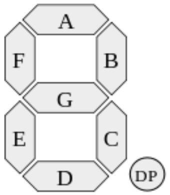
- Print your initials
- *Optional* extras
  - ❑ Calculator with Hex keys



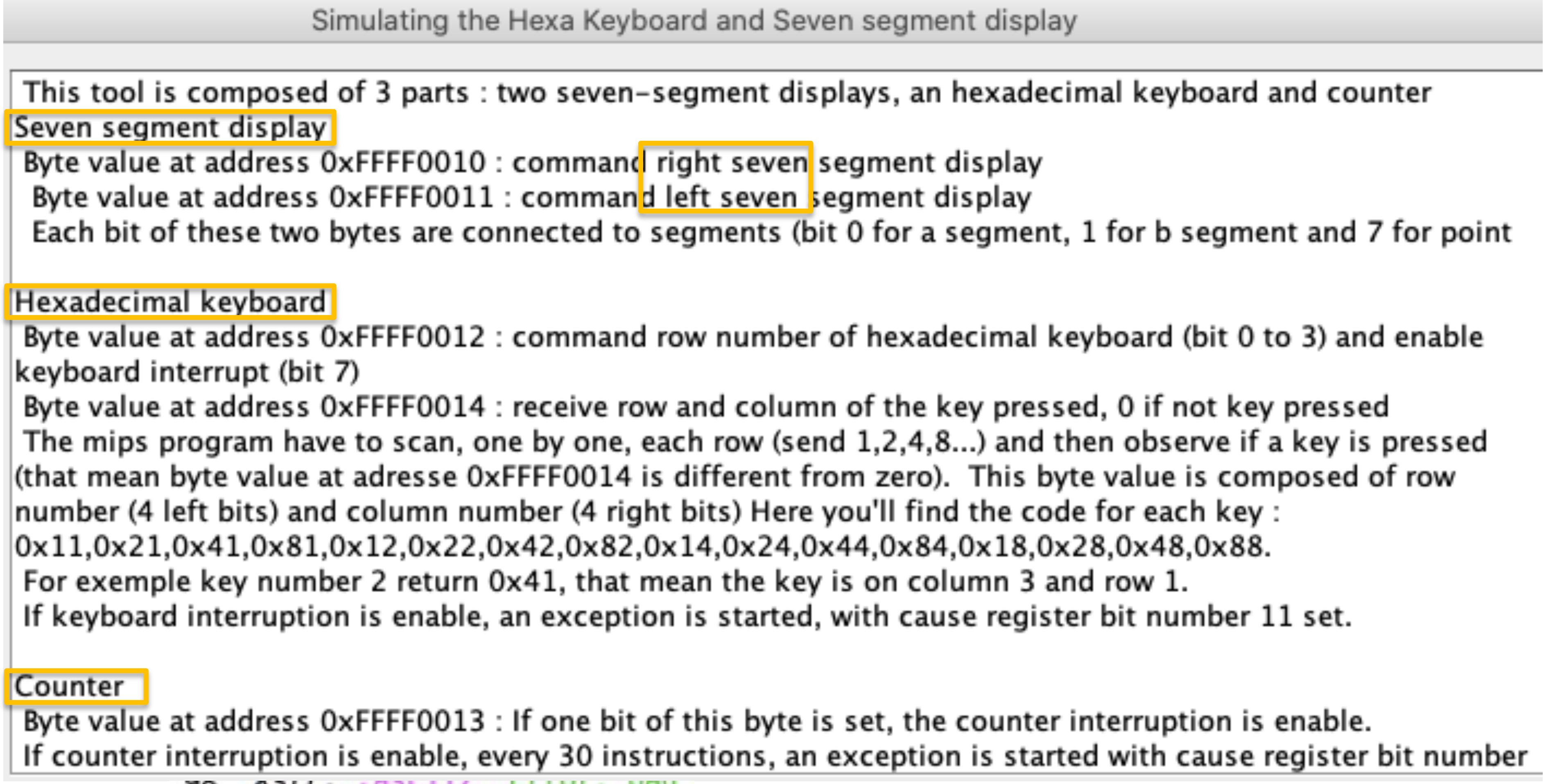
Hexadecimal encodings for displaying the digits 0 to F

Digit	gfedcba	abcdefg	a	b	c	d	e	f	g
0	0x3F	0x7E	on	on	on	on	on	on	off
1	0x06	0x30	off	on	on	off	off	off	off
2	0x5B	0x6D	on	on	off	on	on	off	on
3	0x4F	0x79	on	on	on	on	off	off	on
4	0x66	0x33	off	on	on	off	off	on	on
5	0x6D	0x5B	on	off	on	on	off	on	on
6	0x7D	0x5F	on	off	on	on	on	on	on
7	0x07	0x70	on	on	on	off	off	off	off
8	0x7F	0x7F	on	on	on	on	on	on	on
9	0x6F	0x7B	on	on	on	on	off	on	on
A	0x77	0x77	on	on	on	off	on	on	on
b	0x7C	0x1F	off	off	on	on	on	on	on
C	0x39	0x4E	on	off	off	on	on	on	off
d	0x5E	0x3D	off	on	on	on	on	off	on
E	0x79	0x4F	on	off	off	on	on	on	on
F	0x71	0x47	on	off	off	off	on	on	on





122 Project 2: Digital Lab



Simulating the Hexa Keyboard and Seven segment display

This tool is composed of 3 parts : two seven-segment displays, an hexadecimal keyboard and counter

Seven segment display

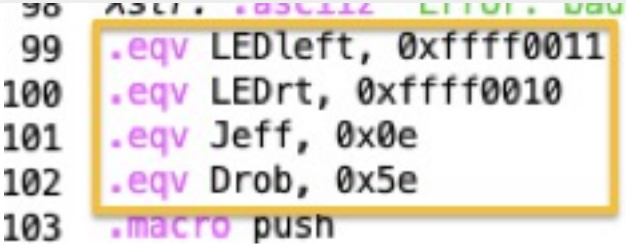
Byte value at address 0xFFFF0010 : command right seven segment display
Byte value at address 0xFFFF0011 : command left seven segment display
Each bit of these two bytes are connected to segments (bit 0 for a segment, 1 for b segment and 7 for point

Hexadecimal keyboard

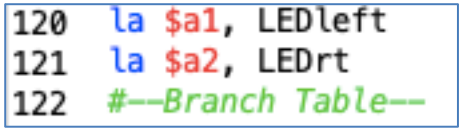
Byte value at address 0xFFFF0012 : command row number of hexadecimal keyboard (bit 0 to 3) and enable keyboard interrupt (bit 7)
Byte value at address 0xFFFF0014 : receive row and column of the key pressed, 0 if not key pressed
The mips program have to scan, one by one, each row (send 1,2,4,8...) and then observe if a key is pressed (that mean byte value at adresse 0xFFFF0014 is different from zero). This byte value is composed of row number (4 left bits) and column number (4 right bits) Here you'll find the code for each key :
0x11,0x21,0x41,0x81,0x12,0x22,0x42,0x82,0x14,0x24,0x44,0x84,0x18,0x28,0x48,0x88.
For exemple key number 2 return 0x41, that mean the key is on column 3 and row 1.
If keyboard interruption is enable, an exception is started, with cause register bit number 11 set.

Counter

Byte value at address 0xFFFF0013 : If one bit of this byte is set, the counter interruption is enable.
If counter interruption is enable, every 30 instructions, an exception is started with cause register bit number



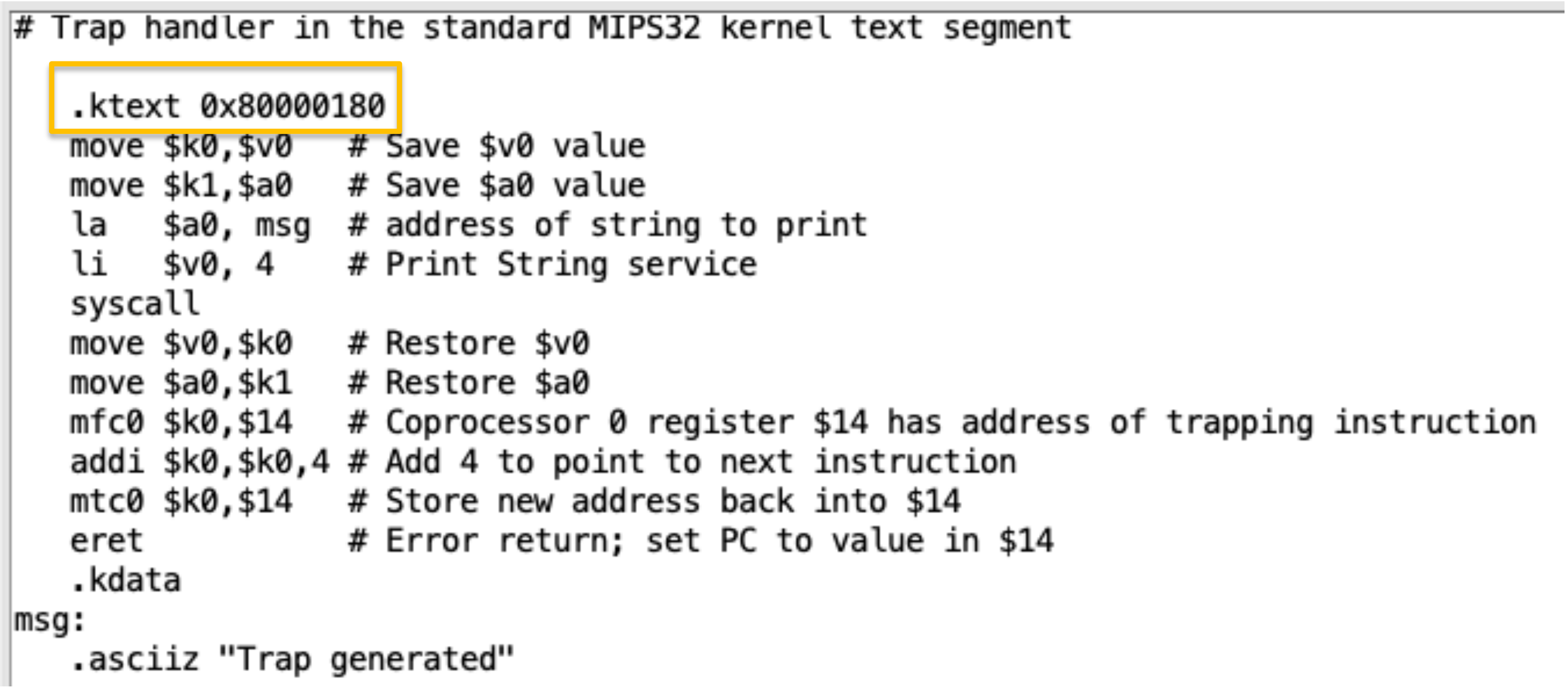
```
.eqv LEDleft, 0xffff0011
.eqv LEDrt, 0xffff0010
.eqv Jeff, 0x0e
.eqv Drob, 0x5e
.macro push
```



```
la $a1, LEDleft
la $a2, LEDrt
#--Branch Table--
```

122 Proj 2 Code: Trap Handler

.ktext



```
# Trap handler in the standard MIPS32 kernel text segment

   .ktext 0x80000180
   move $k0,$v0   # Save $v0 value
   move $k1,$a0   # Save $a0 value
   la   $a0, msg  # address of string to print
   li   $v0, 4    # Print String service
   syscall
   move $v0,$k0   # Restore $v0
   move $a0,$k1   # Restore $a0
   mfc0 $k0,$14   # Coprocessor 0 register $14 has address of trapping instruction
   addi $k0,$k0,4 # Add 4 to point to next instruction
   mtc0 $k0,$14   # Store new address back into $14
   eret           # Error return; set PC to value in $14
   .kdata
msg:
   .asciiz "Trap generated"
```

Exceptions

Exception Handling

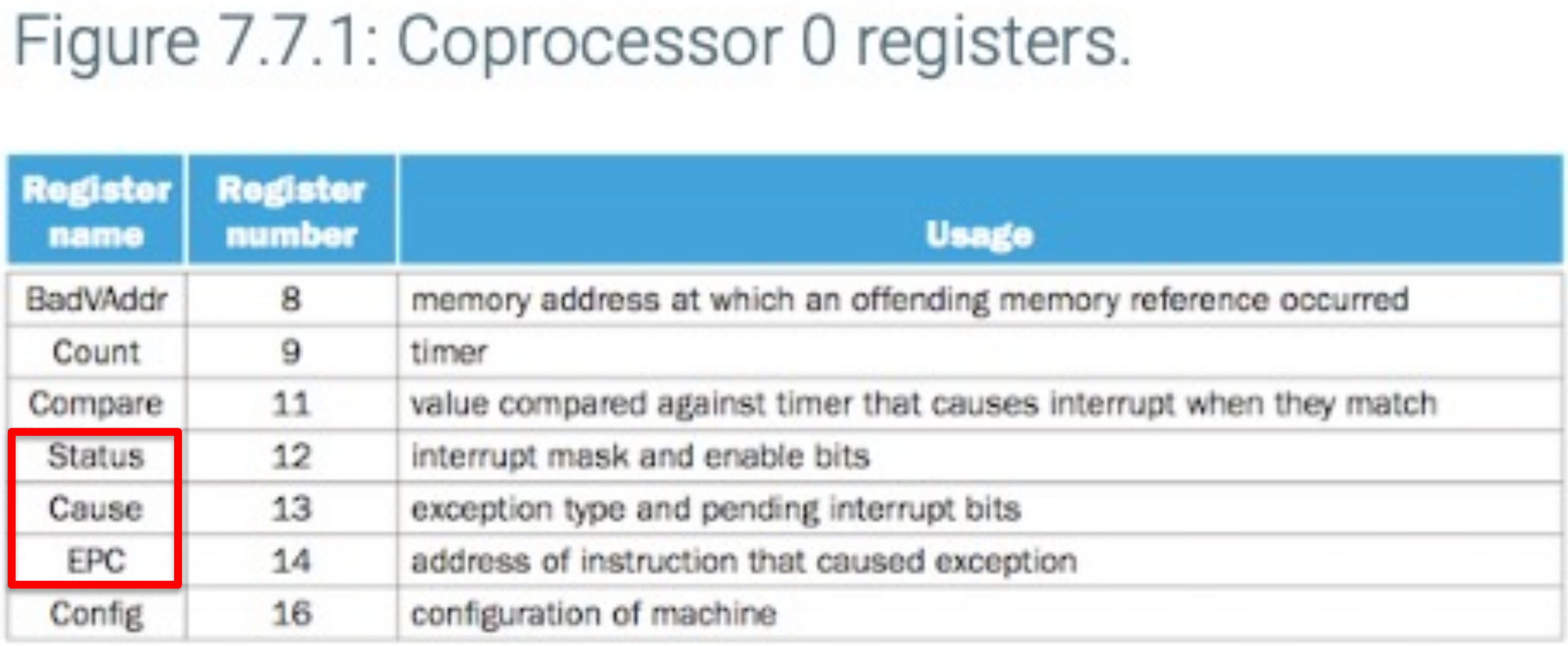


Figure 7.7.1: Coprocessor 0 registers.

Register name	Register number	Usage
BadVAddr	8	memory address at which an offending memory reference occurred
Count	9	timer
Compare	11	value compared against timer that causes interrupt when they match
Status	12	interrupt mask and enable bits
Cause	13	exception type and pending interrupt bits
EPC	14	address of instruction that caused exception
Config	16	configuration of machine

Exception Handling

Figure 7.7.2: The status register (COD Figure A.7.1).

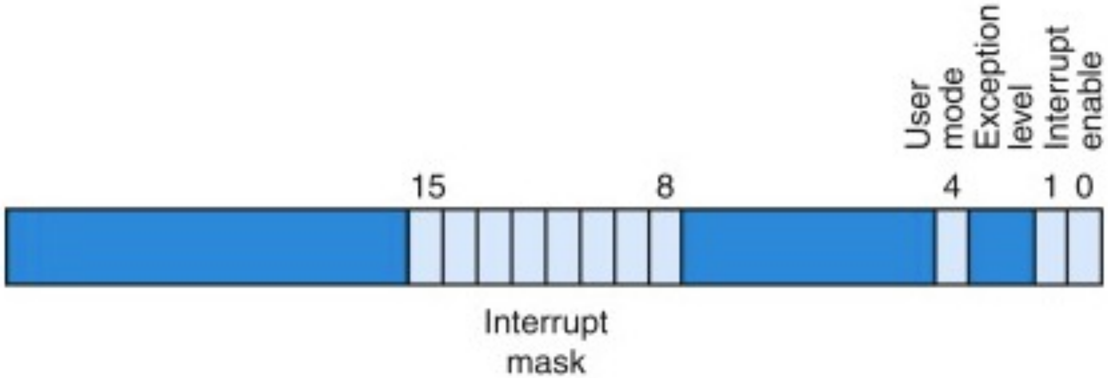
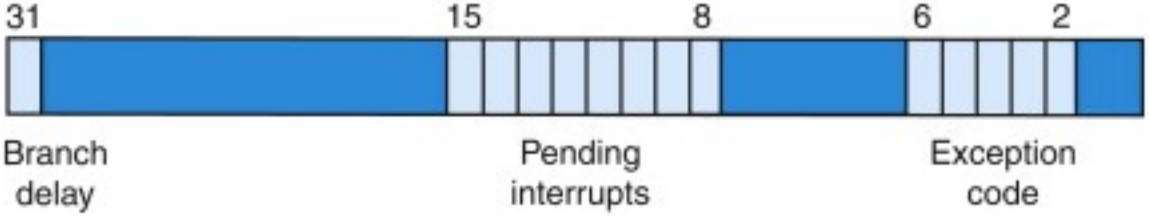


Figure 7.7.3: The cause register (COD Figure A.7.2).



Exception Handling

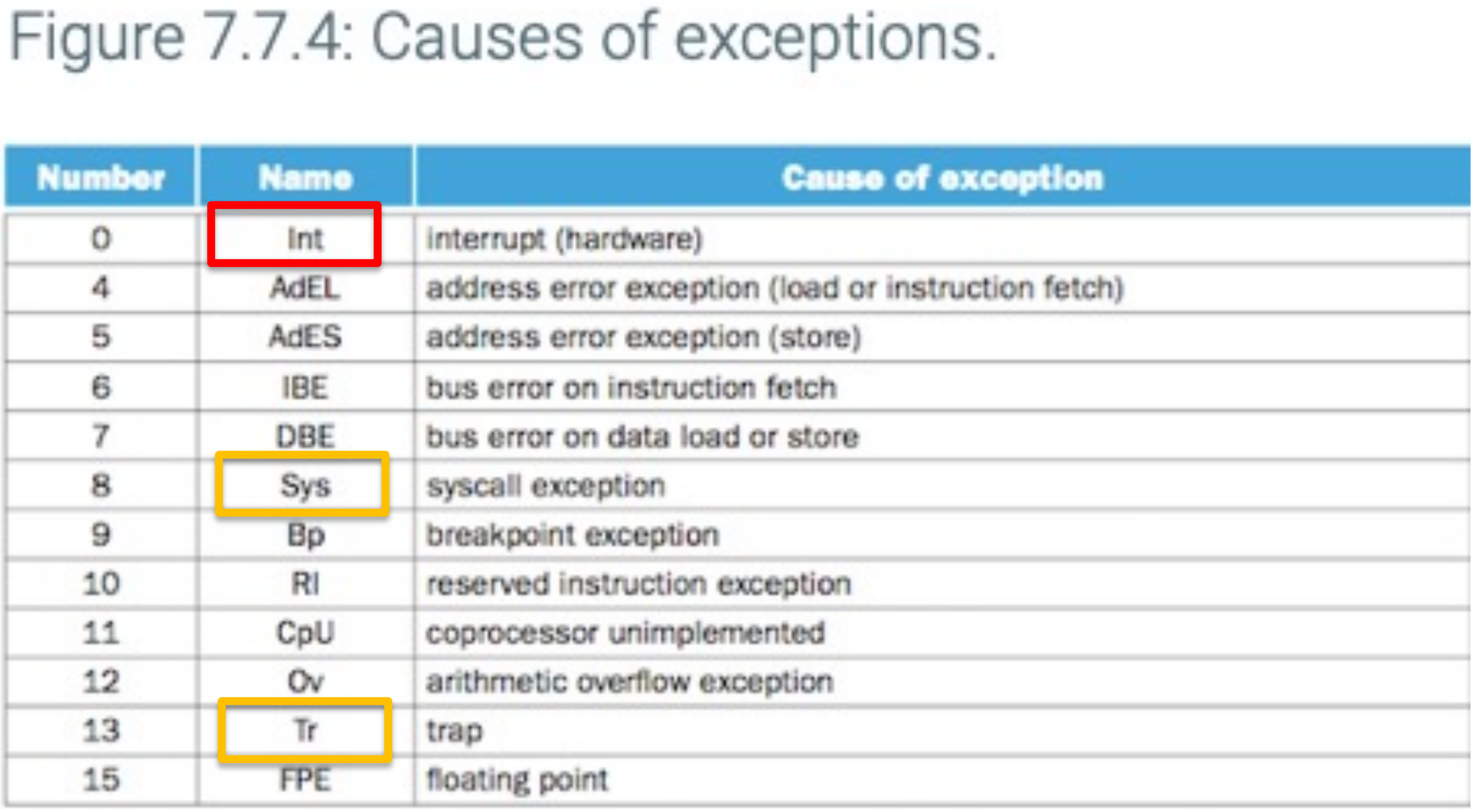
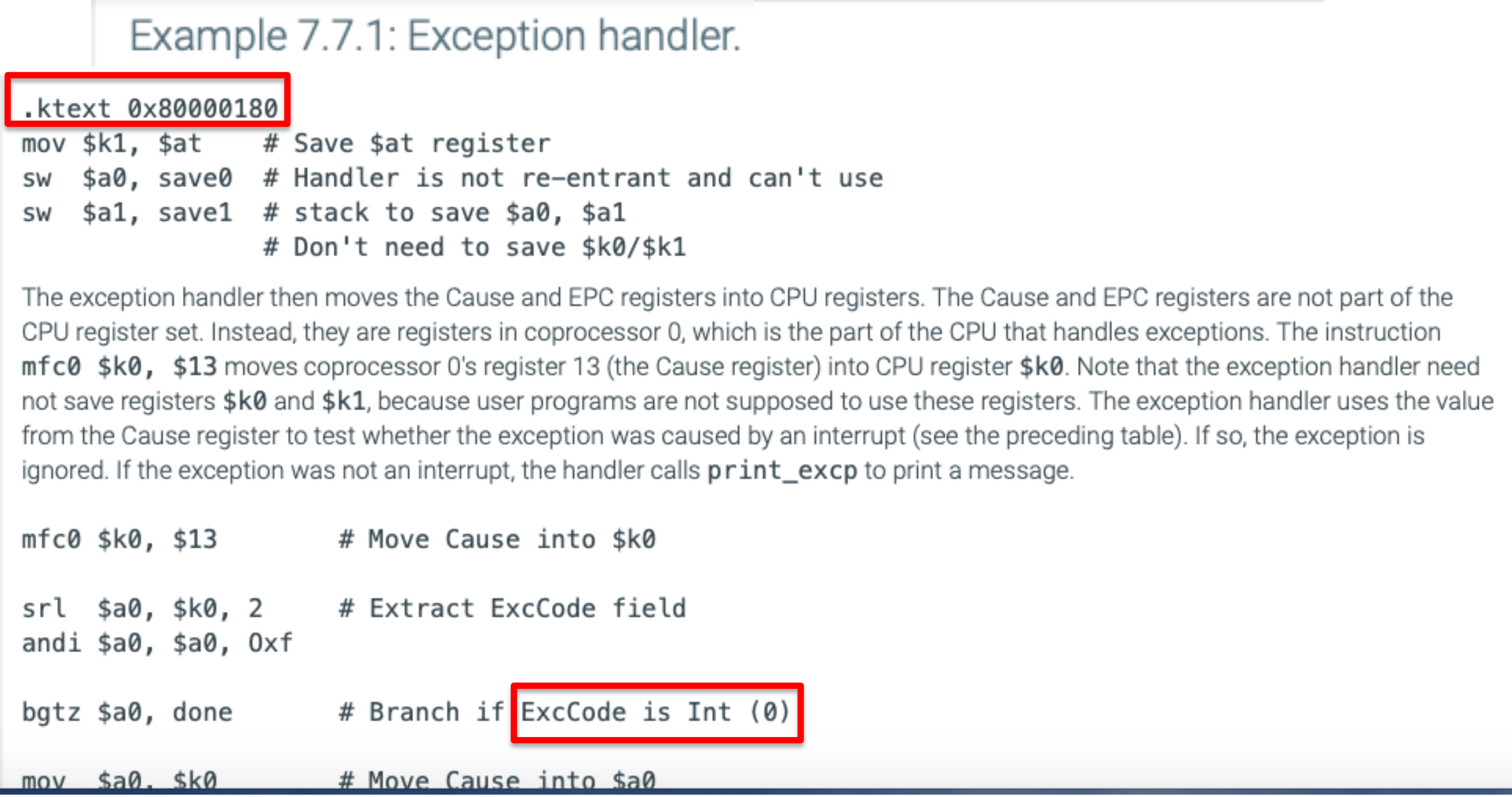


Figure 7.7.4: Causes of exceptions.

Number	Name	Cause of exception
0	Int	interrupt (hardware)
4	AdEL	address error exception (load or instruction fetch)
5	AdES	address error exception (store)
6	IBE	bus error on instruction fetch
7	DBE	bus error on data load or store
8	Sys	syscall exception
9	Bp	breakpoint exception
10	RI	reserved instruction exception
11	CpU	coprocessor unimplemented
12	Ov	arithmetic overflow exception
13	Tr	trap
15	FPE	floating point

Exception Handling



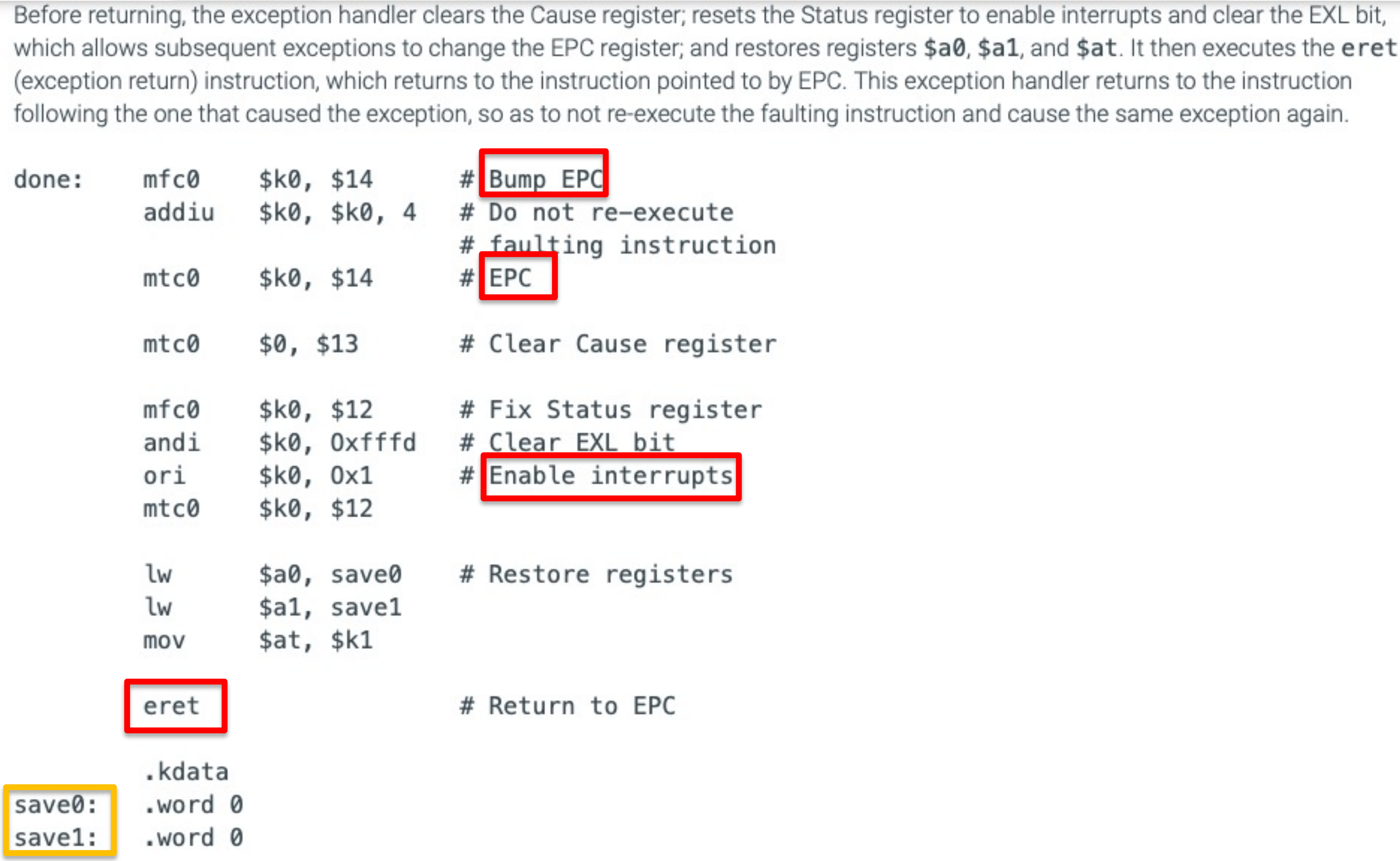
Example 7.7.1: Exception handler.

```
.ktext 0x80000180
mov $k1, $at     # Save $at register
sw  $a0, save0   # Handler is not re-entrant and can't use
sw  $a1, save1   # stack to save $a0, $a1
                 # Don't need to save $k0/$k1
```

The exception handler then moves the Cause and EPC registers into CPU registers. The Cause and EPC registers are not part of the CPU register set. Instead, they are registers in coprocessor 0, which is the part of the CPU that handles exceptions. The instruction `mfc0 $k0, $13` moves coprocessor 0's register 13 (the Cause register) into CPU register `$k0`. Note that the exception handler need not save registers `$k0` and `$k1`, because user programs are not supposed to use these registers. The exception handler uses the value from the Cause register to test whether the exception was caused by an interrupt (see the preceding table). If so, the exception is ignored. If the exception was not an interrupt, the handler calls `print_excp` to print a message.

```
mfc0 $k0, $13        # Move Cause into $k0

srl  $a0, $k0, 2     # Extract ExcCode field
andi $a0, $a0, 0xf

bgtz $a0, done       # Branch if ExcCode is Int (0)

mov  $a0, $k0        # Move Cause into $a0
```

Exception Handling



Before returning, the exception handler clears the Cause register; resets the Status register to enable interrupts and clear the EXL bit, which allows subsequent exceptions to change the EPC register; and restores registers `$a0`, `$a1`, and `$at`. It then executes the `eret` (exception return) instruction, which returns to the instruction pointed to by EPC. This exception handler returns to the instruction following the one that caused the exception, so as to not re-execute the faulting instruction and cause the same exception again.

```
done:   mfc0    $k0, $14       # Bump EPC
        addiu   $k0, $k0, 4    # Do not re-execute
                               # faulting instruction
        mtc0    $k0, $14       # EPC

        mtc0    $0, $13        # Clear Cause register

        mfc0    $k0, $12       # Fix Status register
        andi    $k0, 0xfffd    # Clear EXL bit
        ori     $k0, 0x1       # Enable interrupts
        mtc0    $k0, $12

        lw      $a0, save0     # Restore registers
        lw      $a1, save1
        mov     $at, $k1

        eret                   # Return to EPC

        .kdata
save0:  .word 0
save1:  .word 0
```

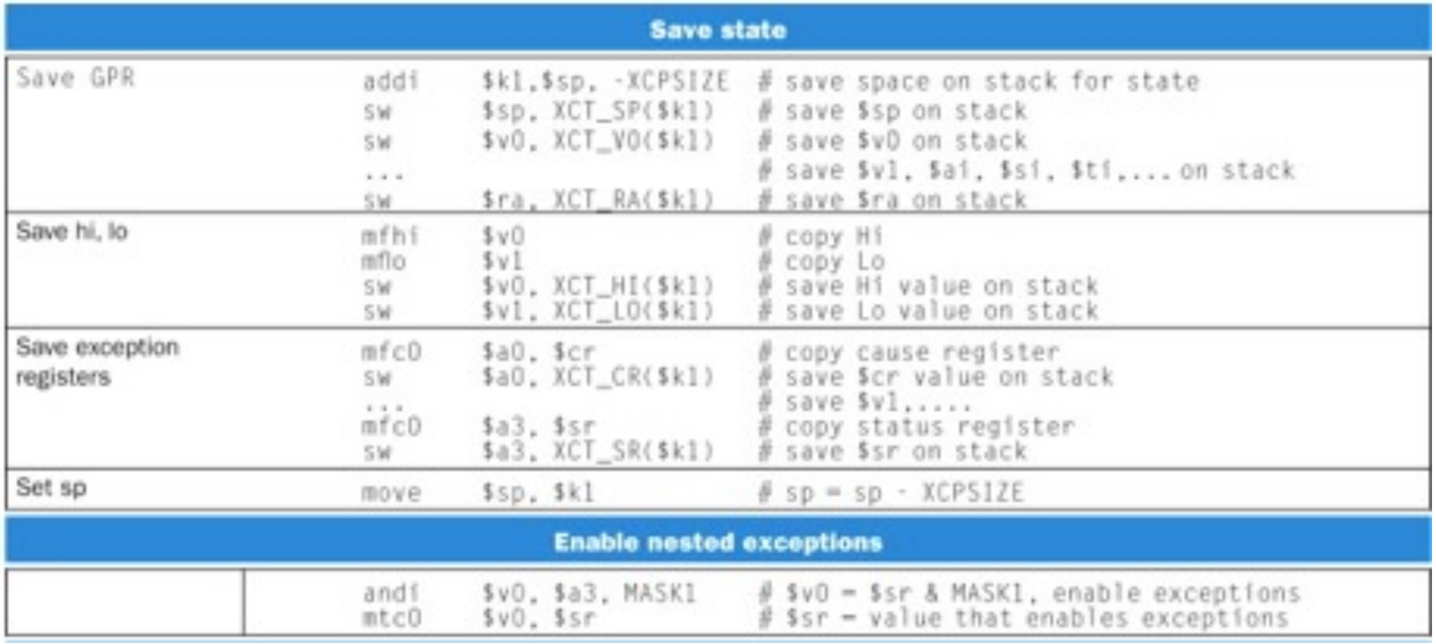
COMP222

Exception Handler

P&H Ch 5

COMP 122: Computer Architecture and Assembly Language
Spring 2020

Figure 5.7.10: MIPS code to save and restore state on an exception (COD Figure 5.34).



Save state

Save GPR	addi	$k1,$sp, -XCPSIZE	# save space on stack for state
	sw	$sp, XCT_SP($k1)	# save $sp on stack
	sw	$v0, XCT_V0($k1)	# save $v0 on stack
	...		# save $v1, $ai, $si, $ti,... on stack
	sw	$ra, XCT_RA($k1)	# save $ra on stack
Save hi, lo	mfhi	$v0	# copy Hi
	mflo	$v1	# copy Lo
	sw	$v0, XCT_HI($k1)	# save Hi value on stack
	sw	$v1, XCT_LO($k1)	# save Lo value on stack
Save exception registers	mfc0	$a0, $cr	# copy cause register
	sw	$a0, XCT_CR($k1)	# save $cr value on stack
	...		# save $v1,....
	mfc0	$a3, $sr	# copy status register
	sw	$a3, XCT_SR($k1)	# save $sr on stack
Set sp	move	$sp, $k1	# sp = sp - XCPSIZE

Enable nested exceptions

	andi	$v0, $a3, MASK1	# $v0 = $sr & MASK1, enable exceptions
	mtc0	$v0, $sr	# $sr = value that enables exceptions

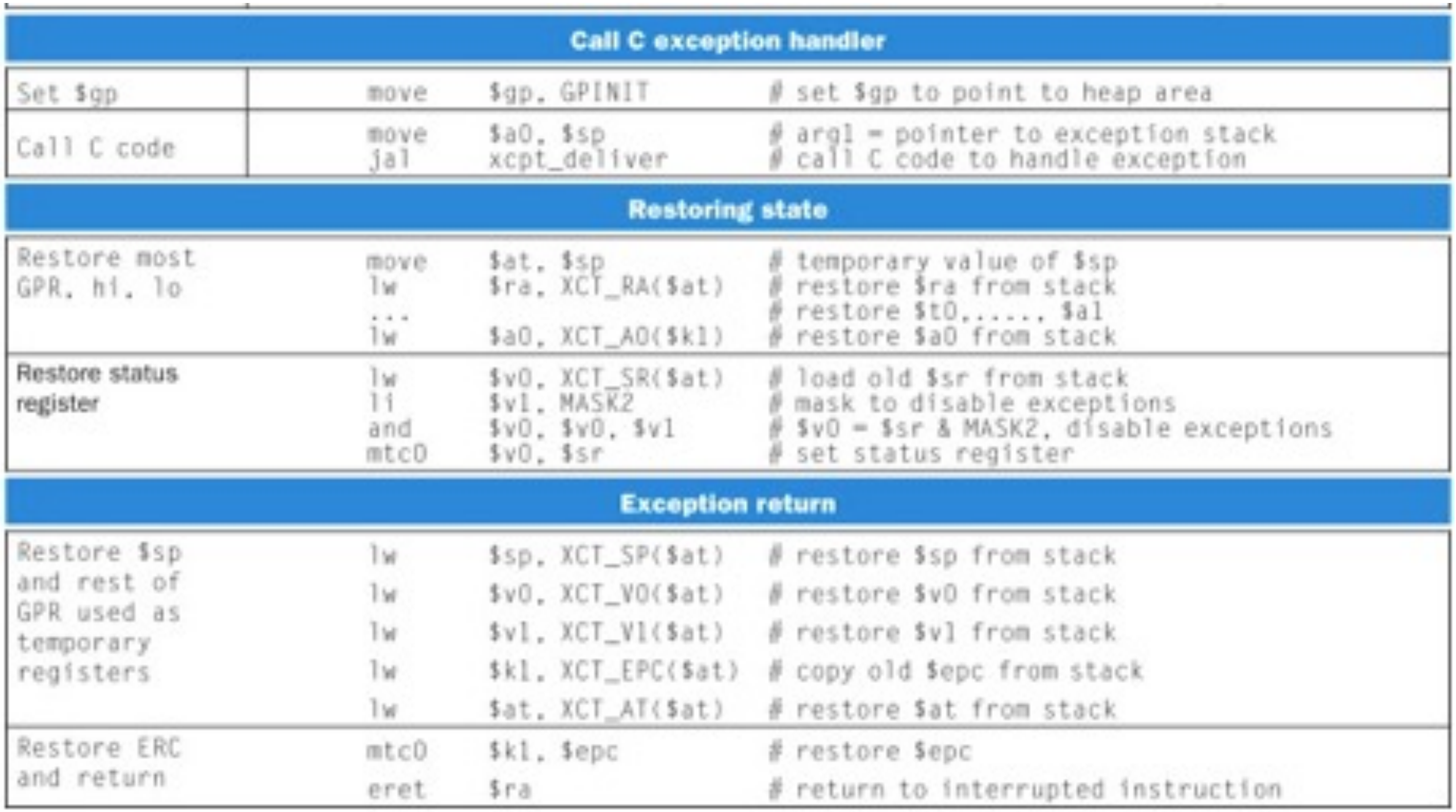
CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP222

Exception Handler

DR JEFF SOFTWARE
INDIE APP DEVELOPER
© Jeff Drobman
2020-23

P&H Ch 5

COMP 122: Computer Architecture and Assembly Language
Spring 2020



Call C exception handler			
Set $gp	move	$gp, GPINIT	# set $gp to point to heap area
Call C code	move jal	$a0, $sp xcpt_deliver	# arg1 = pointer to exception stack # call C code to handle exception
Restoring state			
Restore most GPR, hi, lo	move lw ... lw	$at, $sp $ra, XCT_RA($at) $a0, XCT_A0($k1)	# temporary value of $sp # restore $ra from stack # restore $t0,....., $a1 # restore $a0 from stack
Restore status register	lw li and mtc0	$v0, XCT_SR($at) $v1, MASK2 $v0, $v0, $v1 $v0, $sr	# load old $sr from stack # mask to disable exceptions # $v0 = $sr & MASK2, disable exceptions # set status register
Exception return			
Restore $sp and rest of GPR used as temporary registers	lw lw lw lw lw	$sp, XCT_SP($at) $v0, XCT_V0($at) $v1, XCT_V1($at) $k1, XCT_EPC($at) $at, XCT_AT($at)	# restore $sp from stack # restore $v0 from stack # restore $v1 from stack # copy old $epc from stack # restore $at from stack
Restore ERC and return	mtc0 eret	$k1, $epc $ra	# restore $epc # return to interrupted instruction

System Calls

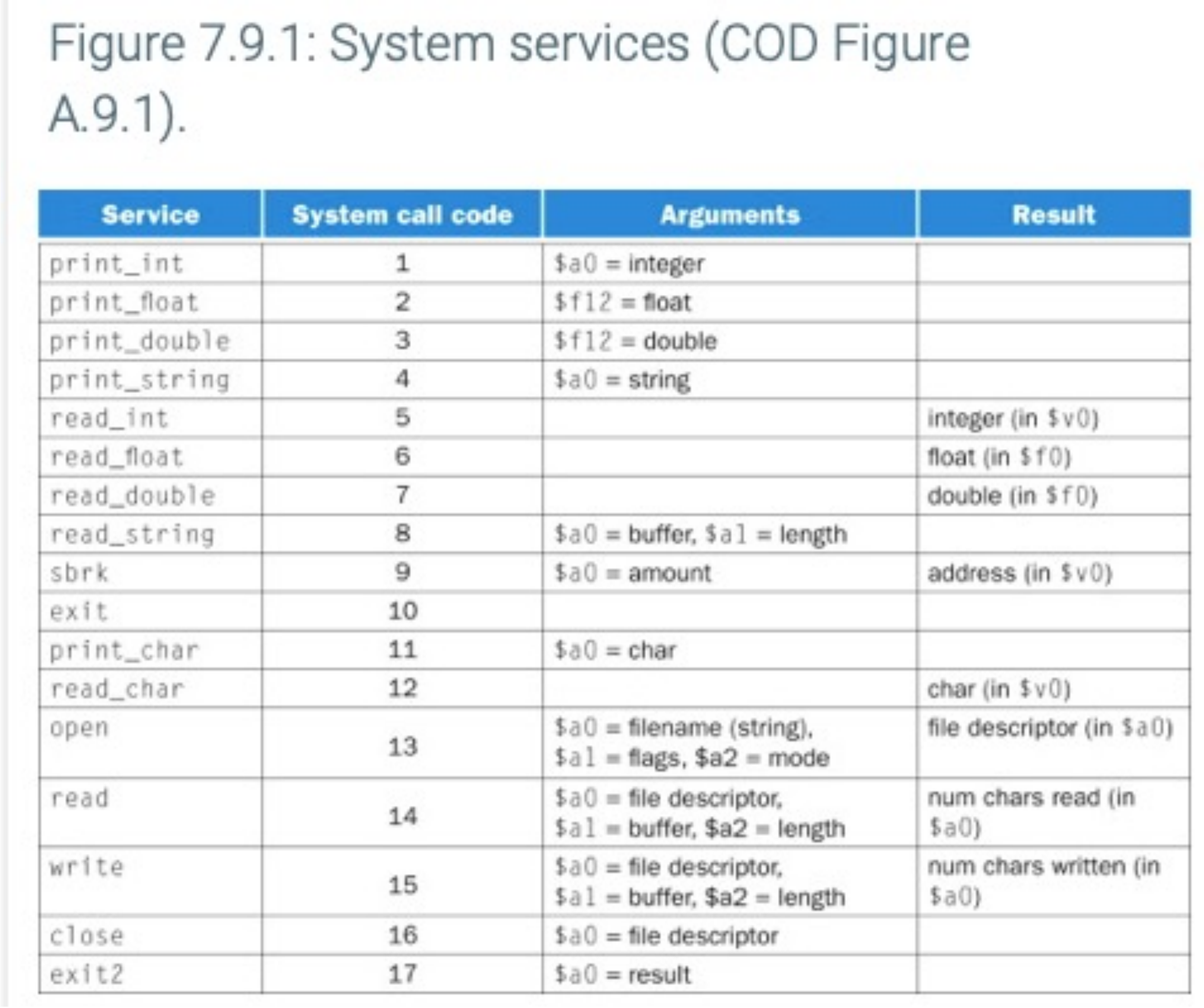
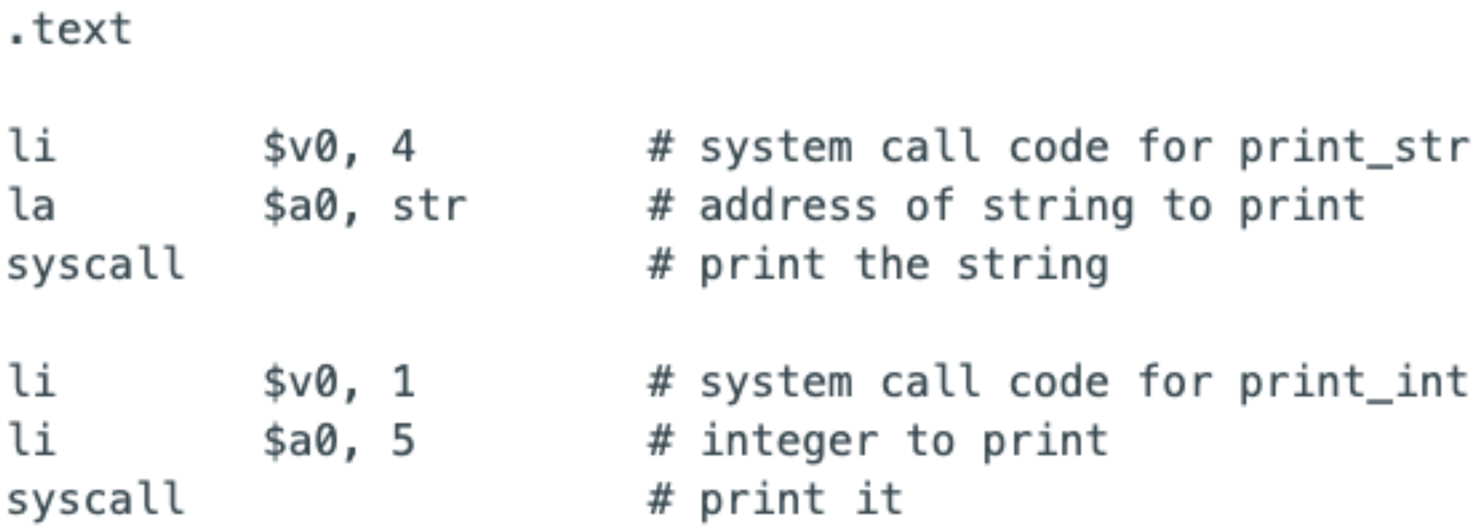


Figure 7.9.1: System services (COD Figure A.9.1).

Service	System call code	Arguments	Result
print_int	1	$a0 = integer	
print_float	2	$f12 = float	
print_double	3	$f12 = double	
print_string	4	$a0 = string	
read_int	5		integer (in $v0)
read_float	6		float (in $f0)
read_double	7		double (in $f0)
read_string	8	$a0 = buffer, $a1 = length	
sbrk	9	$a0 = amount	address (in $v0)
exit	10		
print_char	11	$a0 = char	
read_char	12		char (in $v0)
open	13	$a0 = filename (string), $a1 = flags, $a2 = mode	file descriptor (in $a0)
read	14	$a0 = file descriptor, $a1 = buffer, $a2 = length	num chars read (in $a0)
write	15	$a0 = file descriptor, $a1 = buffer, $a2 = length	num chars written (in $a0)
close	16	$a0 = file descriptor	
exit2	17	$a0 = result	

System Calls: SPIM



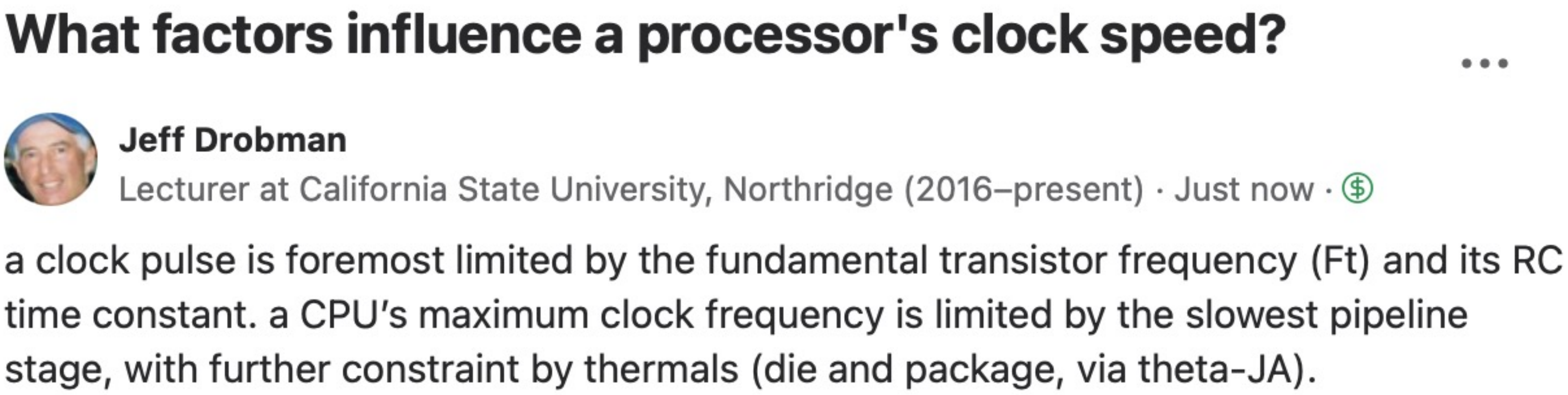
```
.text

li        $v0, 4        # system call code for print_str
la        $a0, str      # address of string to print
syscall                 # print the string

li        $v0, 1        # system call code for print_int
li        $a0, 5        # integer to print
syscall                 # print it
```

Section

Clocks & Cycles

Quora

Clocks



What factors influence a processor's clock speed?

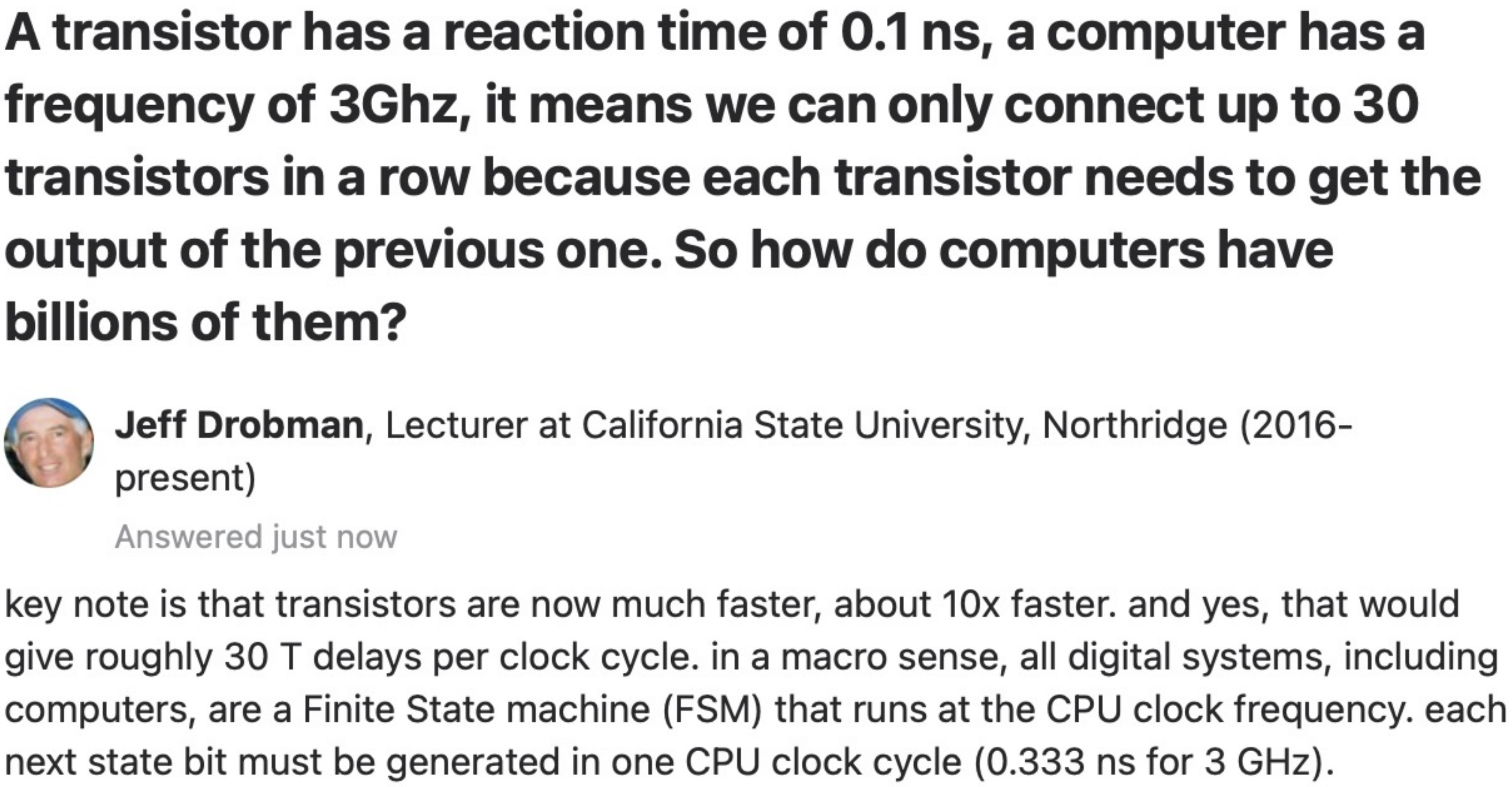
Jeff Drobman

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

a clock pulse is foremost limited by the fundamental transistor frequency (Ft) and its RC time constant. a CPU’s maximum clock frequency is limited by the slowest pipeline stage, with further constraint by thermals (die and package, via theta-JA).

Clock Speeds

Quora



A transistor has a reaction time of 0.1 ns, a computer has a frequency of 3Ghz, it means we can only connect up to 30 transistors in a row because each transistor needs to get the output of the previous one. So how do computers have billions of them?

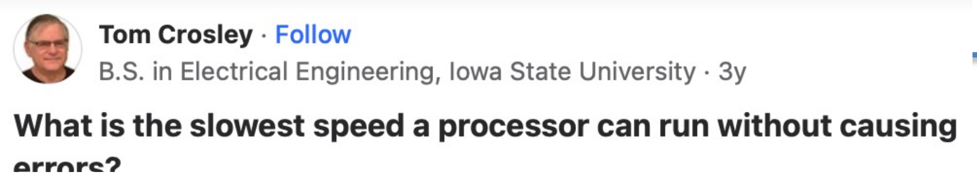
Jeff Drobman, Lecturer at California State University, Northridge (2016-present)

Answered just now

key note is that transistors are now much faster, about 10x faster. and yes, that would give roughly 30 T delays per clock cycle. in a macro sense, all digital systems, including computers, are a Finite State machine (FSM) that runs at the CPU clock frequency. each next state bit must be generated in one CPU clock cycle (0.333 ns for 3 GHz).

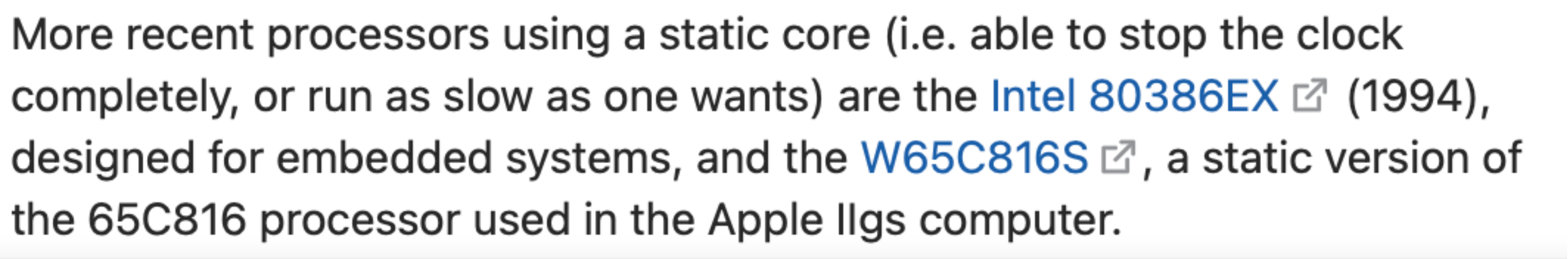
Static CPU

Quora



Tom Crosley · Follow
B.S. in Electrical Engineering, Iowa State University · 3y

What is the slowest speed a processor can run without causing errors?

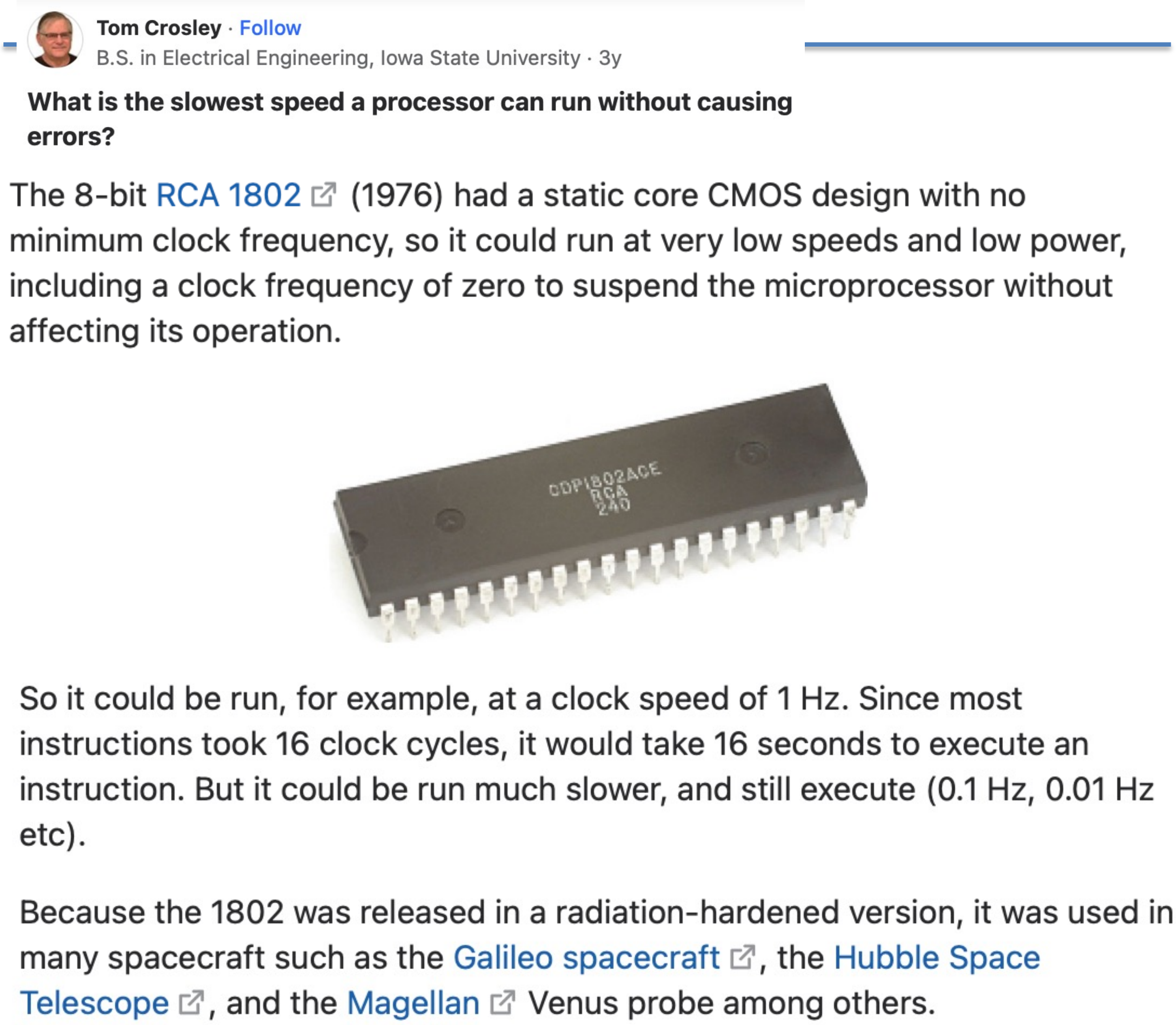


More recent processors using a static core (i.e. able to stop the clock completely, or run as slow as one wants) are the Intel 80386EX (1994), designed for embedded systems, and the W65C816S, a static version of the 65C816 processor used in the Apple IIgs computer.

Slow Clocks

Quora

Static CPU



Tom Crosley · Follow
B.S. in Electrical Engineering, Iowa State University · 3y

What is the slowest speed a processor can run without causing errors?

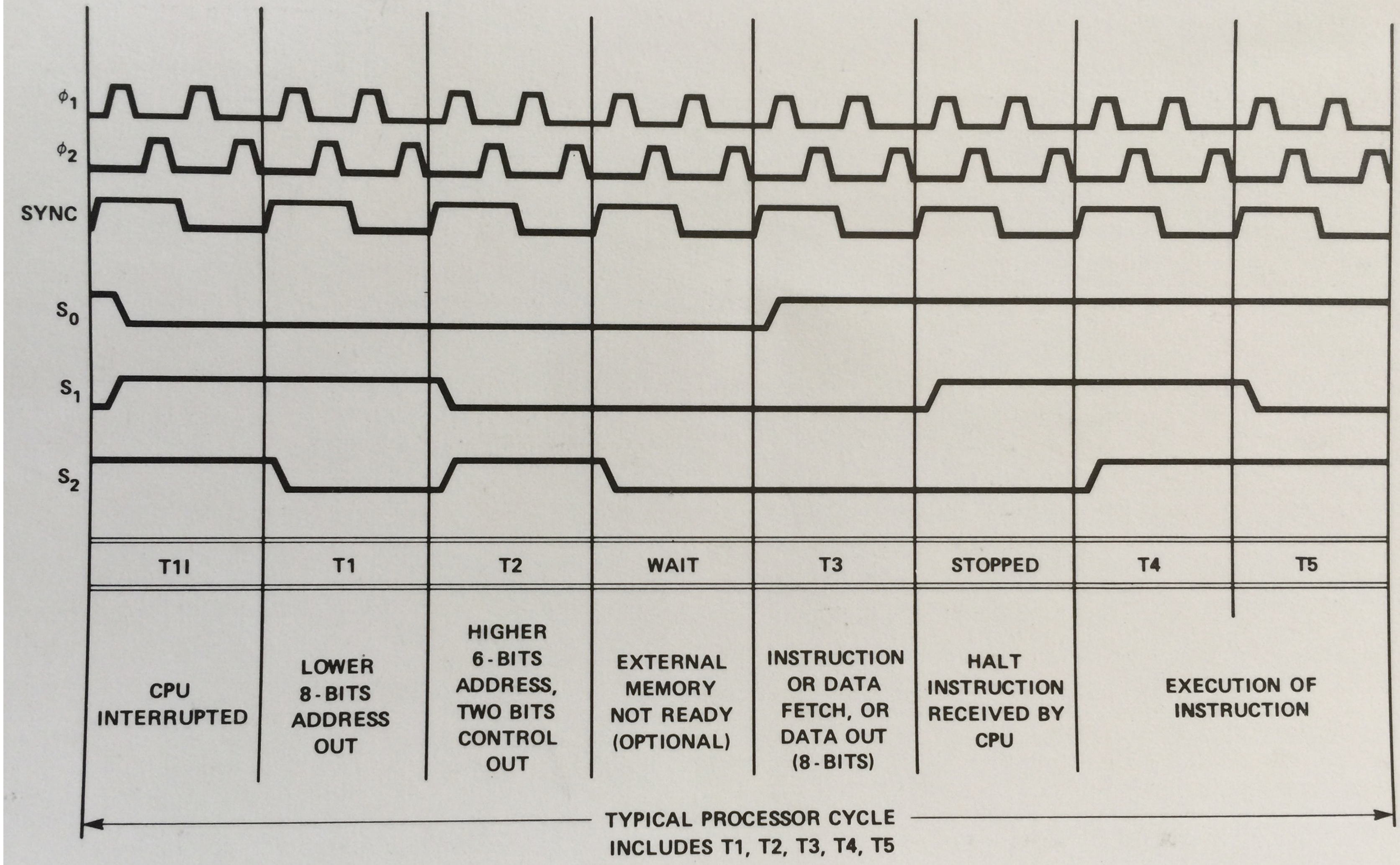
The 8-bit RCA 1802 (1976) had a static core CMOS design with no minimum clock frequency, so it could run at very low speeds and low power, including a clock frequency of zero to suspend the microprocessor without affecting its operation.

So it could be run, for example, at a clock speed of 1 Hz. Since most instructions took 16 clock cycles, it would take 16 seconds to execute an instruction. But it could be run much slower, and still execute (0.1 Hz, 0.01 Hz etc).

Because the 1802 was released in a radiation-hardened version, it was used in many spacecraft such as the Galileo spacecraft, the Hubble Space Telescope, and the Magellan Venus probe among others.

CISC Instruction Cycle

MCS-8

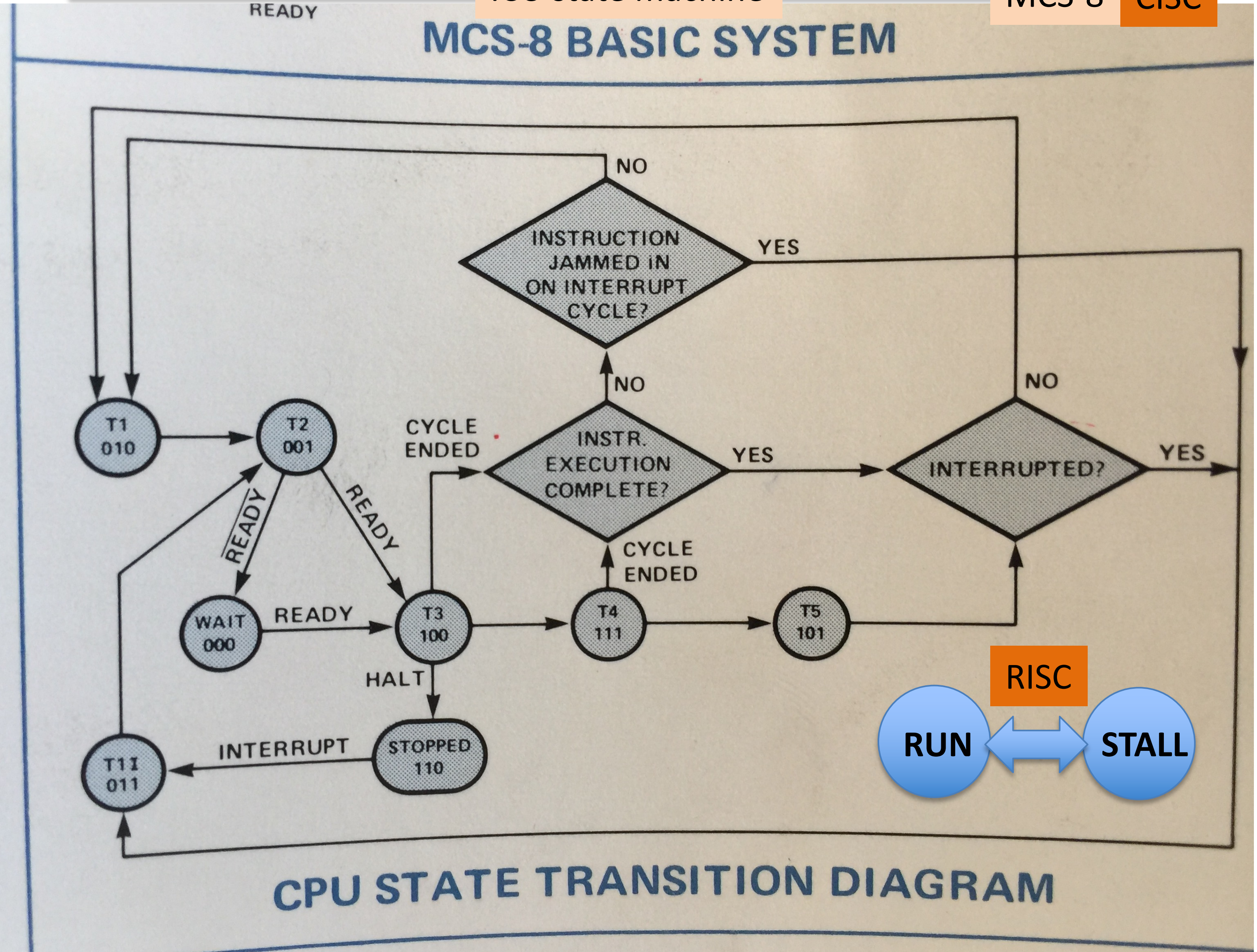

ϕ_1
ϕ_2
SYNC
S_0
S_1
S_2
T1I
T1
T2
WAIT
T3
STOPPED
T4
T5
CPU INTERRUPTED
LOWER 8-BITS ADDRESS OUT
HIGHER 6-BITS ADDRESS, TWO BITS CONTROL OUT
EXTERNAL MEMORY NOT READY (OPTIONAL)
INSTRUCTION OR DATA FETCH, OR DATA OUT (8-BITS)
HALT INSTRUCTION RECEIVED BY CPU
EXECUTION OF INSTRUCTION
TYPICAL PROCESSOR CYCLE INCLUDES T1, T2, T3, T4, T5

MCS-8 BASIC INSTRUCTION CYCLE

CISC State Diagram

ICU state machine

MCS-8 CISC



RISC

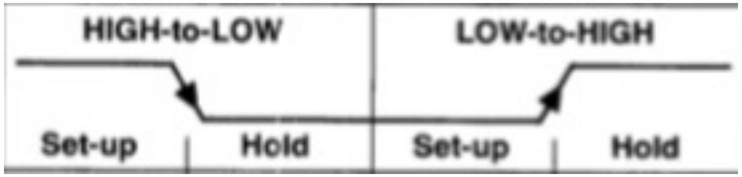
Instruction Cycles

Clock Sync

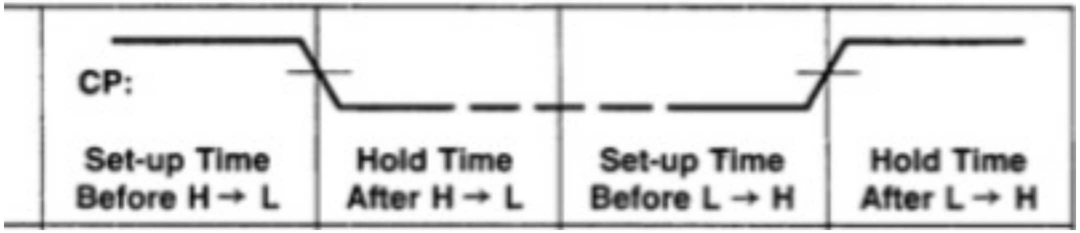
Clocks ⇔ Cycles

clock speed is normally the frequency that a CPU operates at, and inversely, defines the ***clock period*** or ***cycle time***.

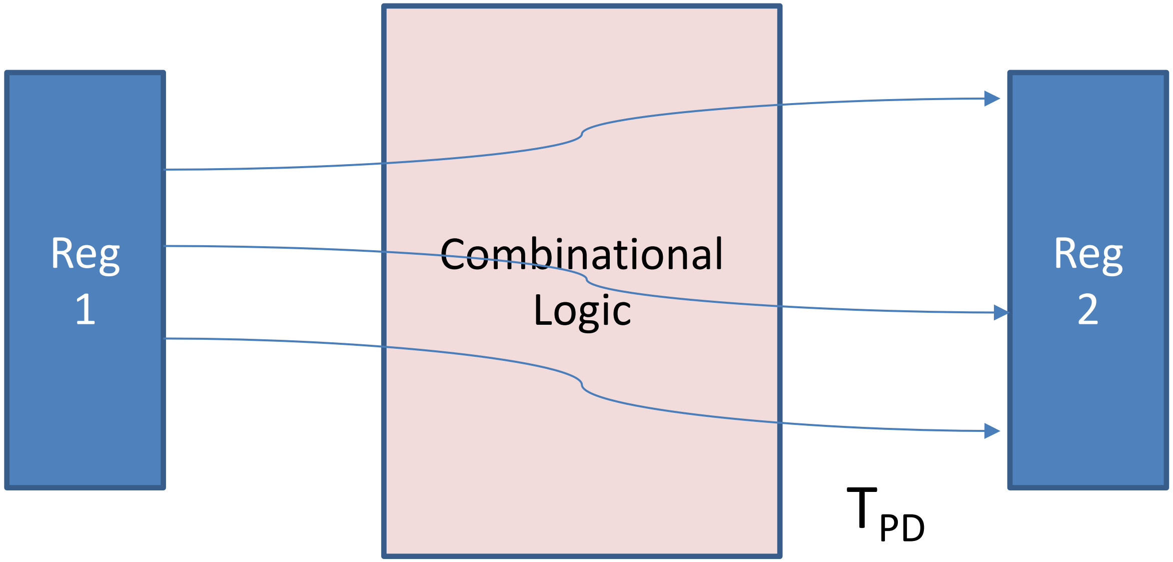
CPI (*clocks* per instruction) is the average number of *cycles* it takes to execute an instruction – per a given *instruction stream*.



Set-up and Hold Times Relative to Clock (CP) Input.



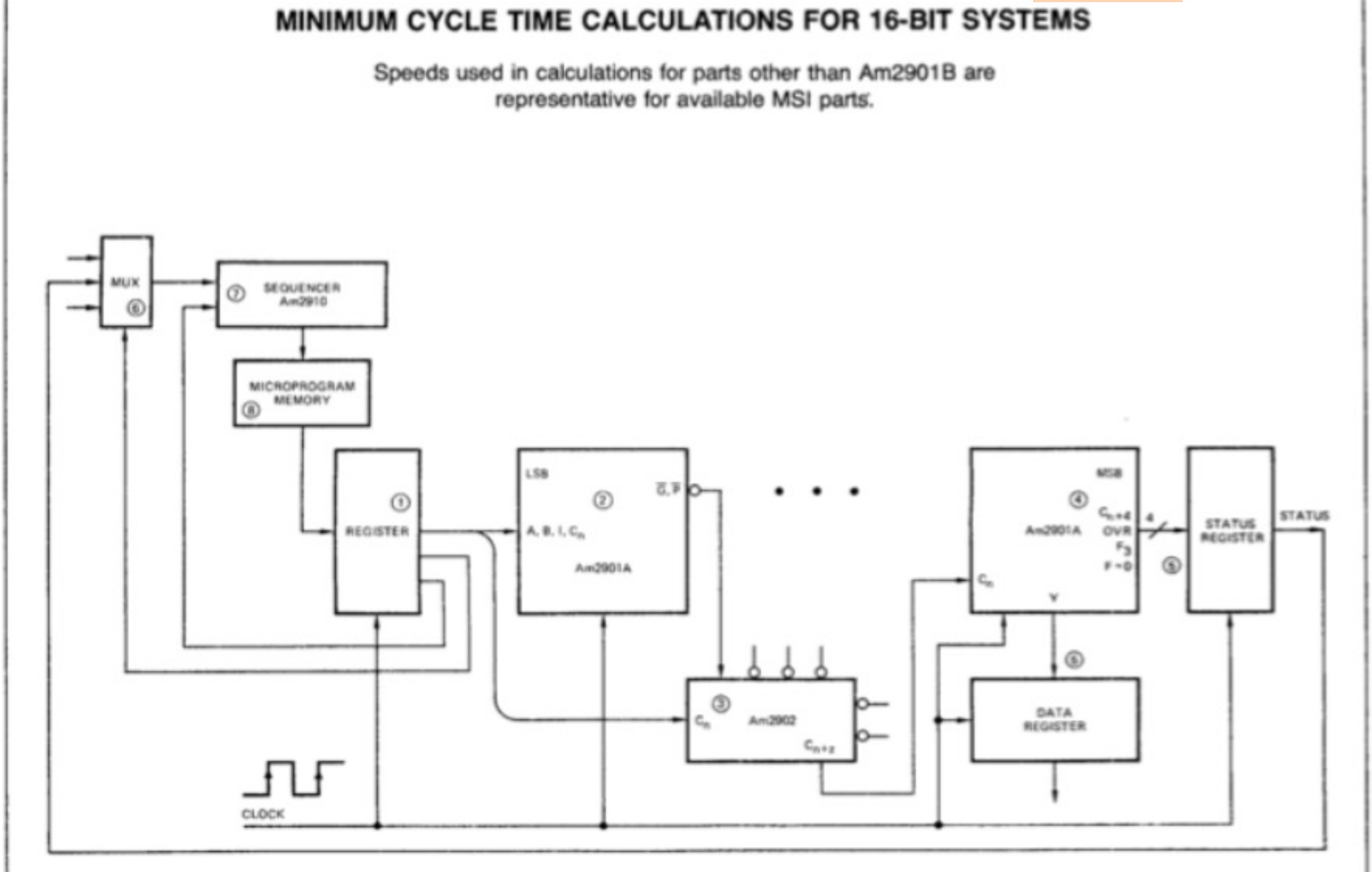
Cycle Time Limit: Slowest Path



$$T_{CYC} = T_{CQ1} - T_{PD} - T_{SU2}$$

Cycle Times

Am2900

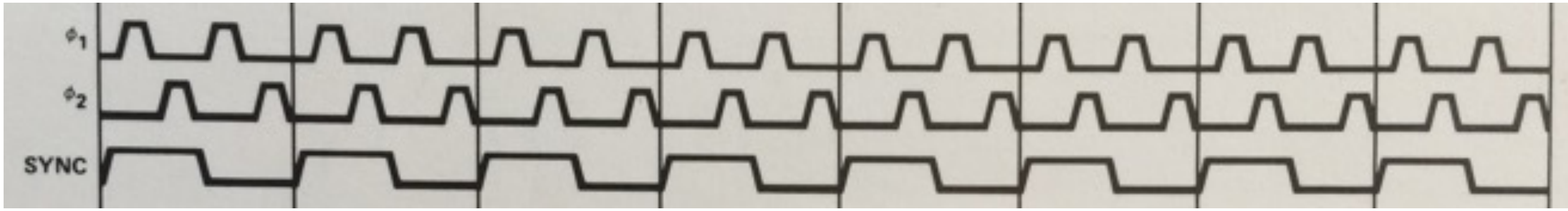


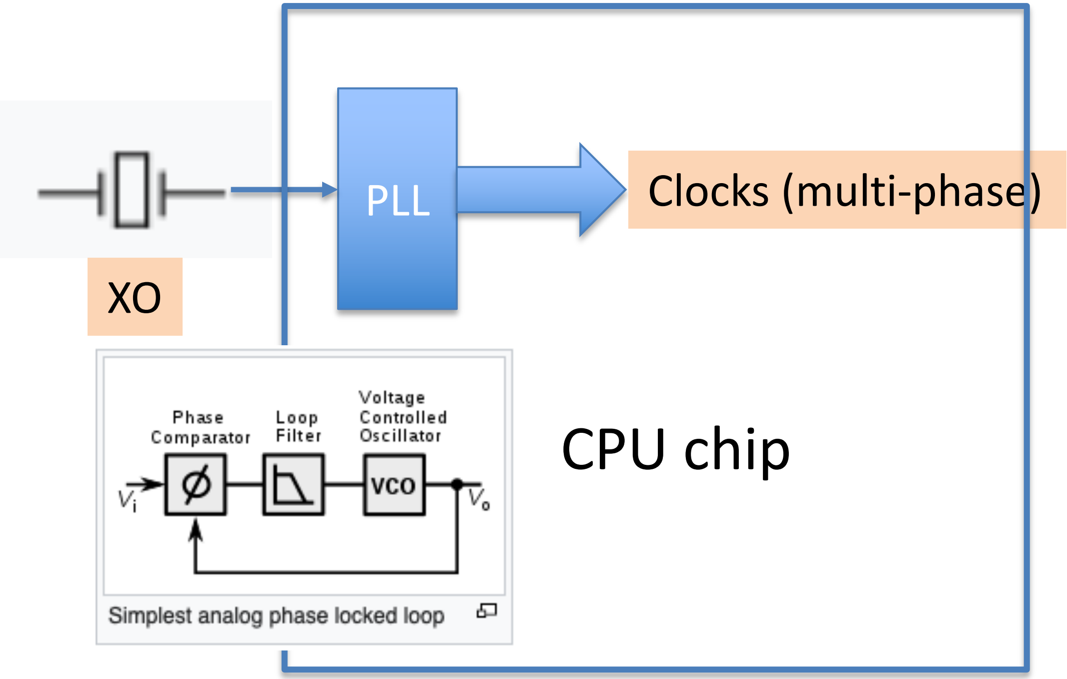
MINIMUM CYCLE TIME CALCULATIONS FOR 16-BIT SYSTEMS

Speeds used in calculations for parts other than Am2901B are representative for available MSI parts.

CPU Chip Clock Source

2-phase clock




XO

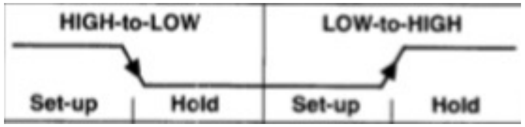
PLL

Clocks (multi-phase)

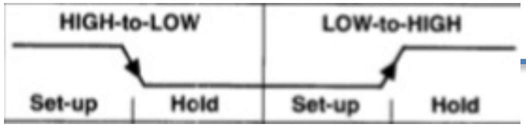
CPU chip

Simplest analog phase locked loop

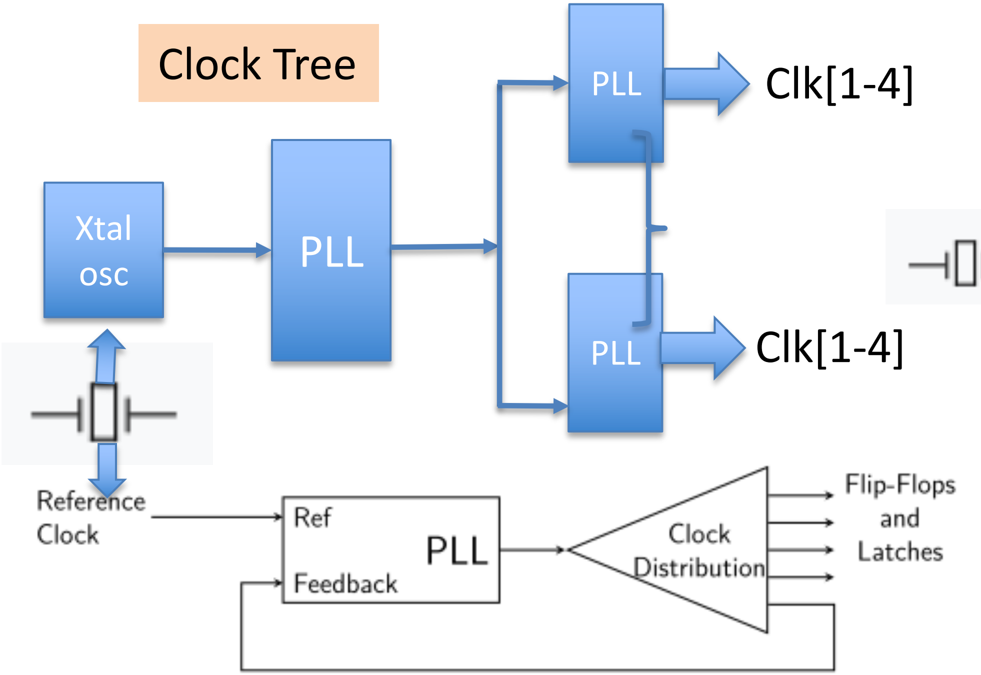
PLL will *multiply*
300 MHz xtal freq
Up to 4 GHz



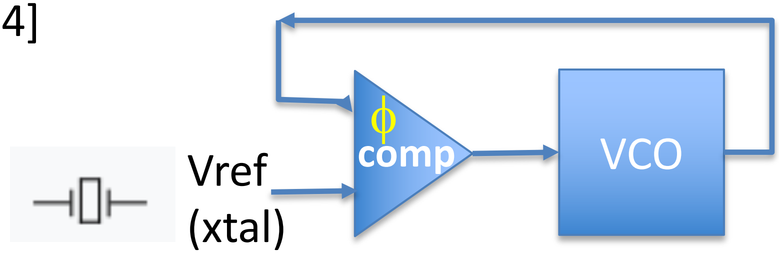
Clock Gen

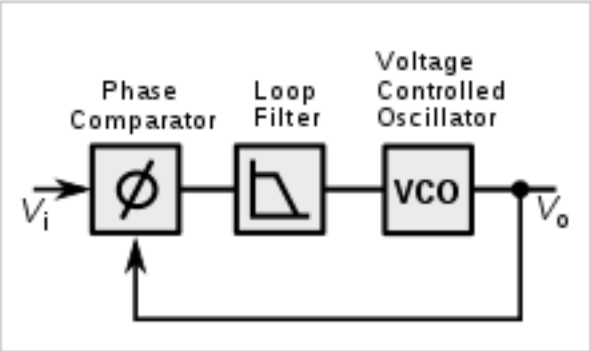


Clock Tree

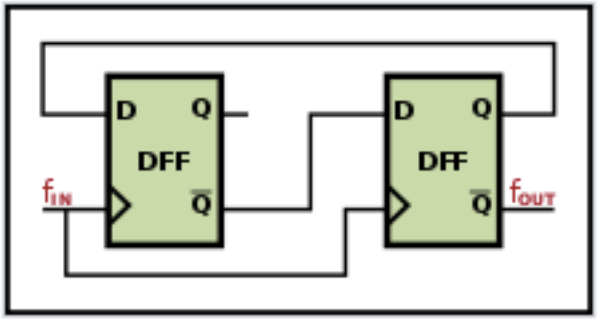


PLL= ***Phase*** Locked Loop





Simplest analog phase locked loop



An example digital divider (by 4) for use in the feedback path of a multiplying PLL

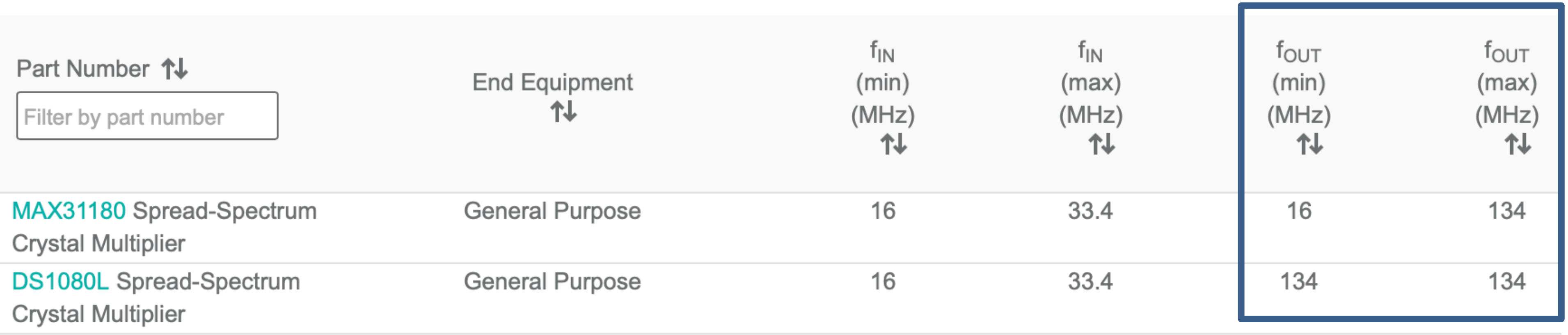
- Jitter (ppm)
- Wander (ppm/day)

Freq divider (by 4)

ON Semiconductor HC4046A

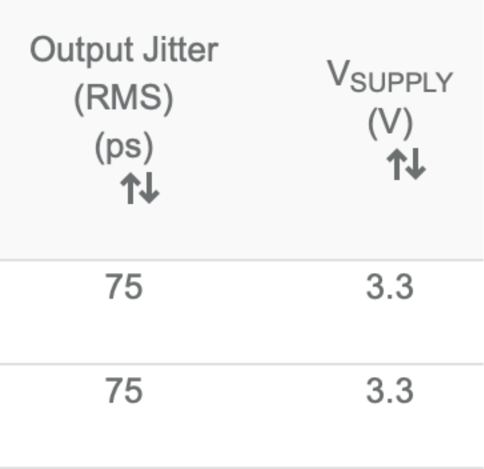
Maxim/ADI Clock Gen

maxim integrated™ | NOW PART OF ANALOG DEVICES



Part Number ⇅ Filter by part number	End Equipment ⇅	f_{IN} (min) (MHz) ⇅	f_{IN} (max) (MHz) ⇅	f_{OUT} (min) (MHz) ⇅	f_{OUT} (max) (MHz) ⇅
MAX31180 Spread-Spectrum Crystal Multiplier	General Purpose	16	33.4	16	134
DS1080L Spread-Spectrum Crystal Multiplier	General Purpose	16	33.4	134	134

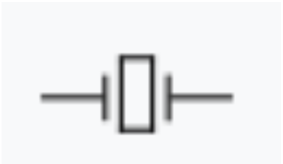
Jitter

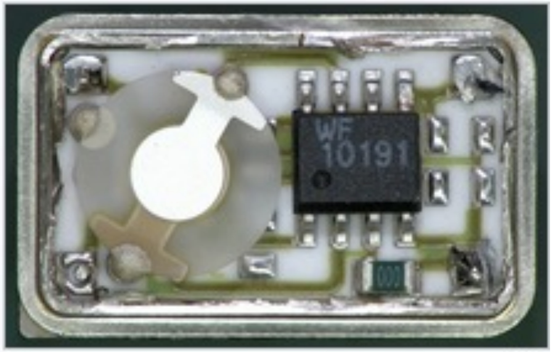


Output Jitter (RMS) (ps) ⇅	V_{SUPPLY} (V) ⇅
75	3.3
75	3.3

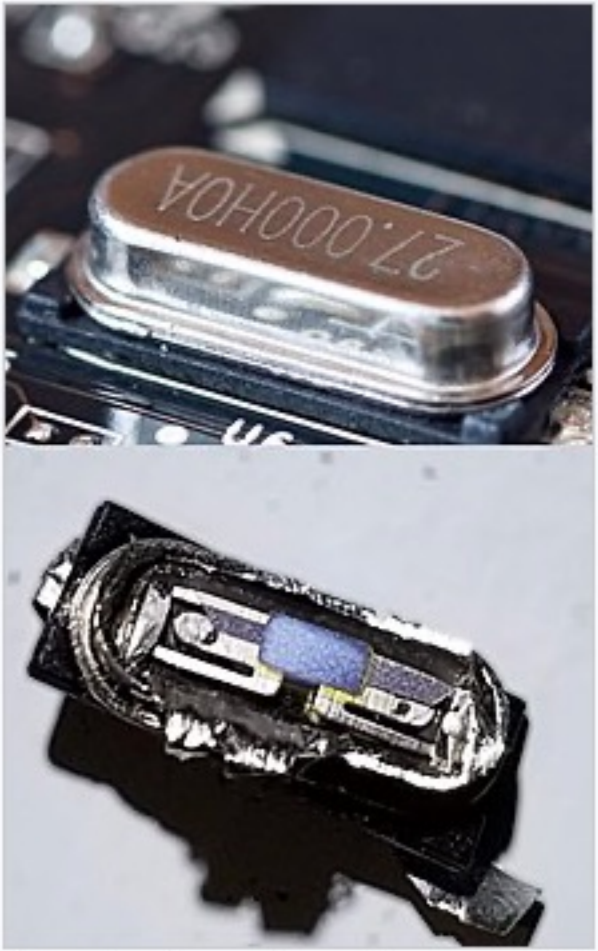
Clock Source: Xtal Osc

Crystal oscillator





Inside a modern DIP package quartz crystal oscillator module.It includes a ceramic PCB base, oscillator, divider chip (/8), bypass capacitor, and an AT cut crystal.



Internals of a quartz crystal.

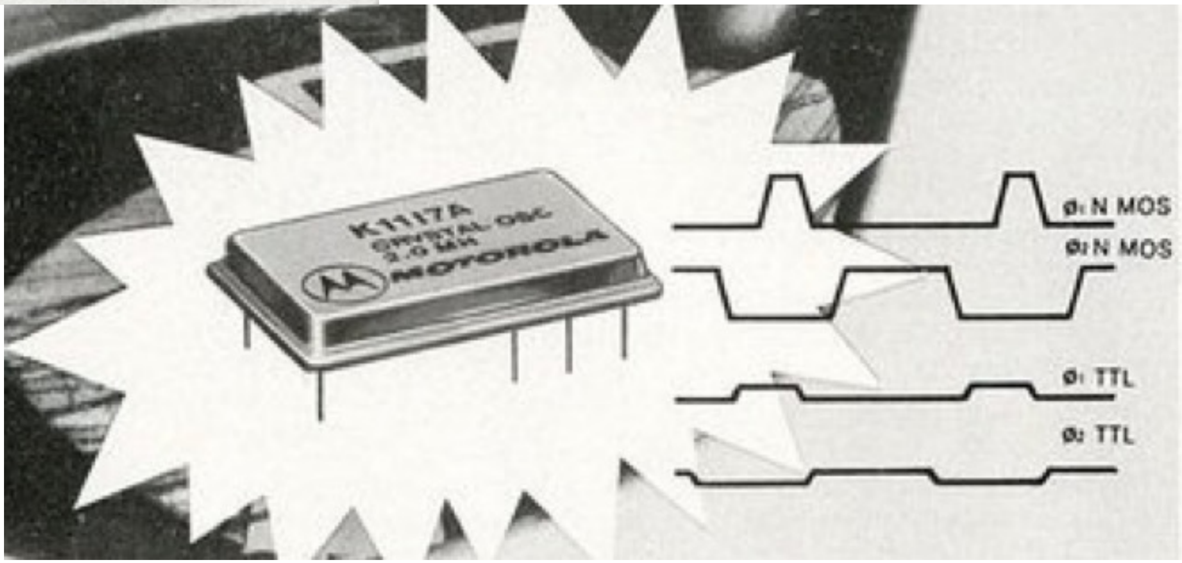
Cluster of natural quartz crystals

A synthetic quartz crystal grown by the hydrothermal synthesis, about 19 cm long and weighing about 127 g

Old MPU ∅ Clock Gen

i8080 MC6800

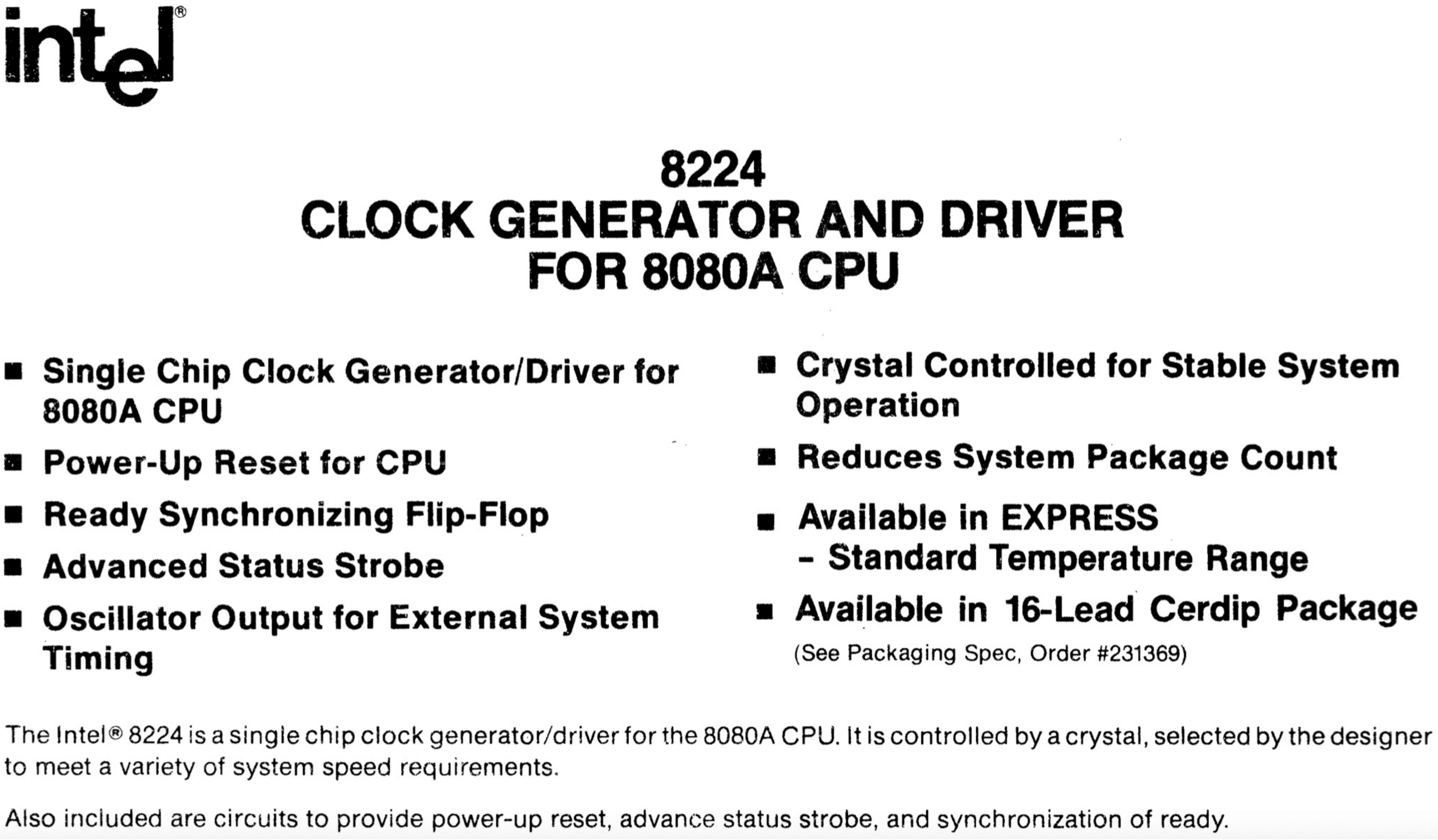
How to drive a microprocessor.



For the Intel 8080 CPU, use our K1117A.
For the Motorola MC6800 MPU, use our MC6870A, 71A or 71B.

Old MPU ⌀ Clock Gen

i8080 i8224



intel®

8224 CLOCK GENERATOR AND DRIVER FOR 8080A CPU

- Single Chip Clock Generator/Driver for 8080A CPU
- Power-Up Reset for CPU
- Ready Synchronizing Flip-Flop
- Advanced Status Strobe
- Oscillator Output for External System Timing
- Crystal Controlled for Stable System Operation
- Reduces System Package Count
- Available in EXPRESS
  - Standard Temperature Range
- Available in 16-Lead Cerdip Package

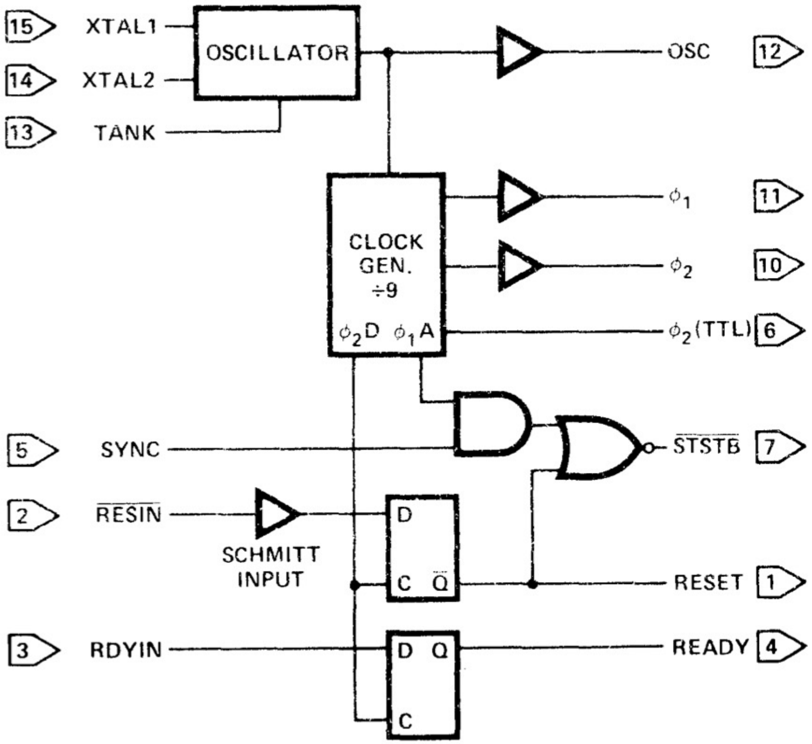
(See Packaging Spec, Order #231369)

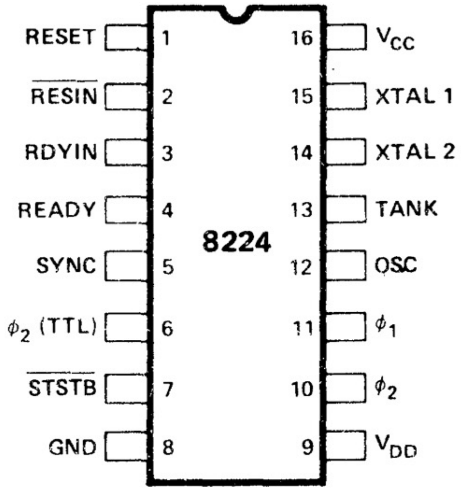
The Intel® 8224 is a single chip clock generator/driver for the 8080A CPU. It is controlled by a crystal, selected by the designer to meet a variety of system speed requirements.

Also included are circuits to provide power-up reset, advance status strobe, and synchronization of ready.

Old MPU ⌀ Clock Gen

i8080 i8224





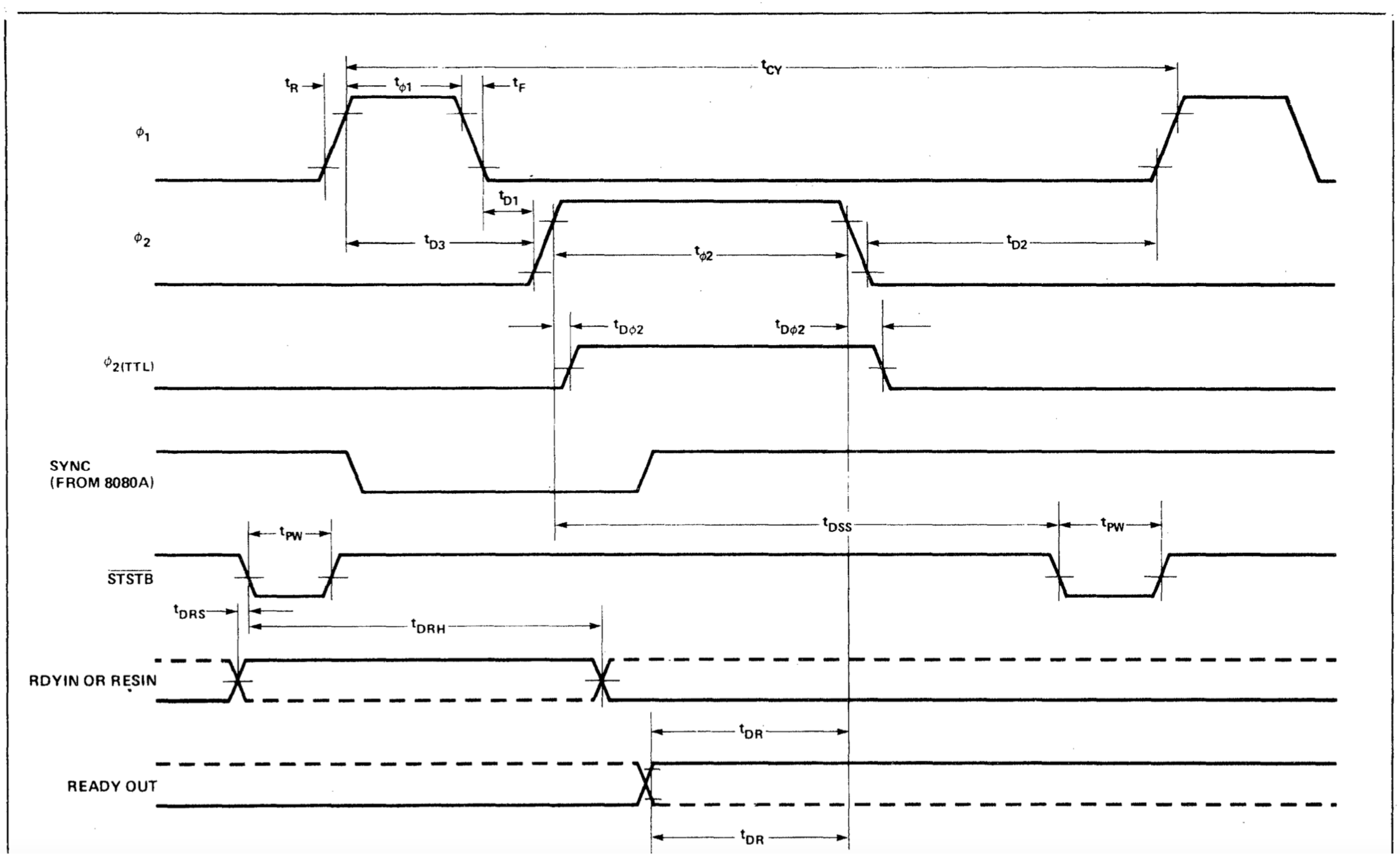
$\overline{RESIN}$	RESET INPUT
RESET	RESET OUTPUT
RDYIN	READY INPUT
READY	READY OUTPUT
SYNC	SYNC INPUT
$\overline{STSTB}$	STATUS STB (ACTIVE LOW)
ϕ_1	8080
ϕ_2	CLOCKS

XTAL 1	CONNECTIONS
XTAL 2	FOR CRYSTAL
TANK	USED WITH OVERTONE XTAL
OSC	OSCILLATOR OUTPUT
ϕ_2 (TTL)	ϕ_2 CLK (TTL LEVEL)
V_{CC}	+5V
V_{DD}	+12V
GND	0V

Old MPU ⌀ Clock Gen

i8080 i8224

WAVEFORMS

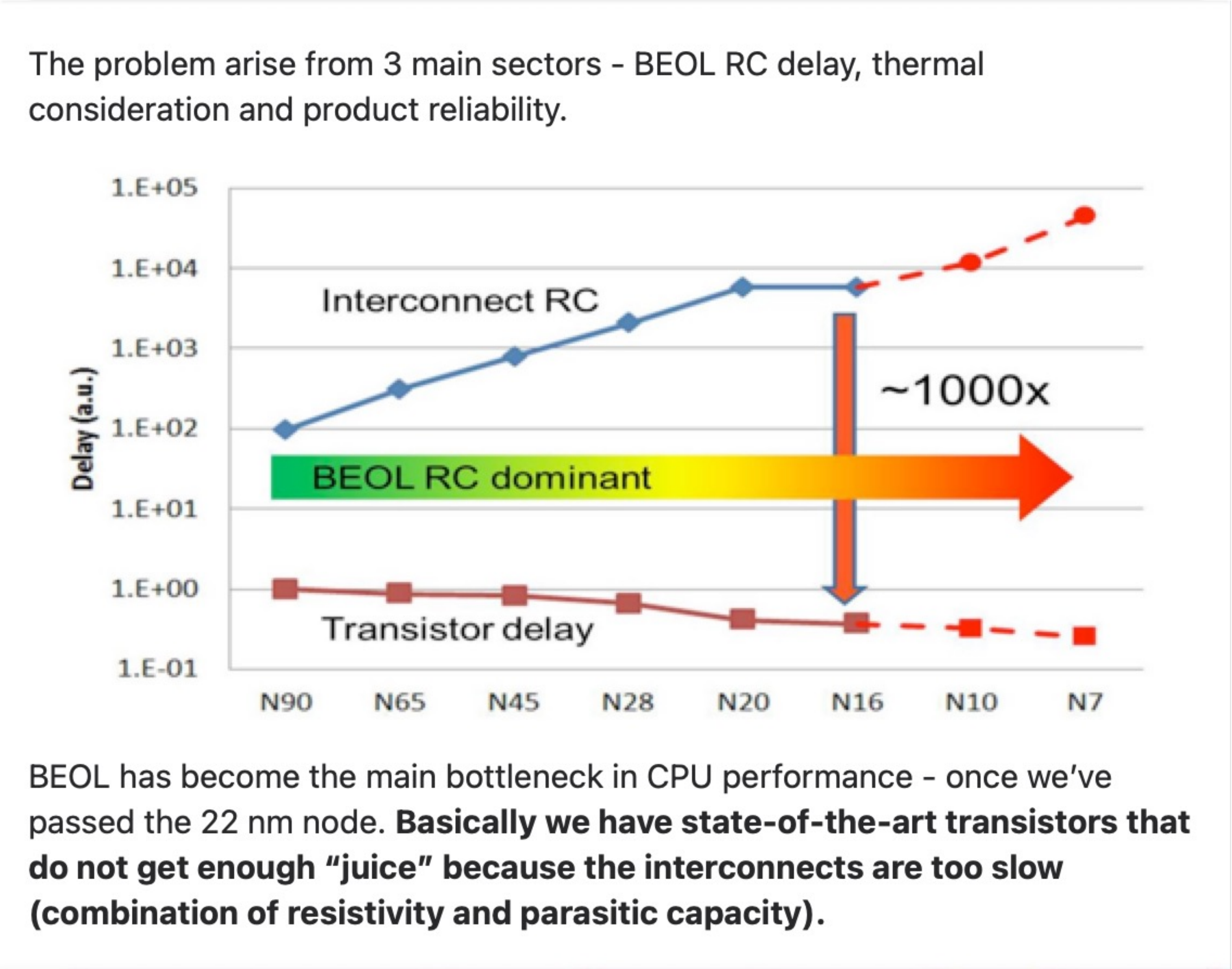


Clock Source: Xtal Osc

Crystal oscillator types and their abbreviations:

- **ATCXO** — Analog temperature controlled crystal oscillator
- **CDXO** — Calibrated dual crystal oscillator
- **DTCXO** — Digital temperature compensated crystal oscillator
- **EMXO** — Evacuated miniature crystal oscillator
- **GPSDO** — Global positioning system disciplined oscillator
- **MCXO** — Microcomputer-compensated crystal oscillator
- **OCVCXO** — oven-controlled voltage-controlled crystal oscillator
- **OCXO** — Oven-controlled crystal oscillator
- **RbXO** — Rubidium crystal oscillators (RbXO), a crystal oscillator (can be an save power
- **TCVCXO** — Temperature-compensated voltage-controlled crystal oscillator
- **TCXO** — Temperature-compensated crystal oscillator
- **TMXO** – Tactical miniature crystal oscillator[67]
- **TSXO** — Temperature-sensing crystal oscillator, an adaptation of the TCXO
- **VCTCXO** — Voltage-controlled temperature-compensated crystal oscillator
- **VCXO** — Voltage-controlled crystal oscillator

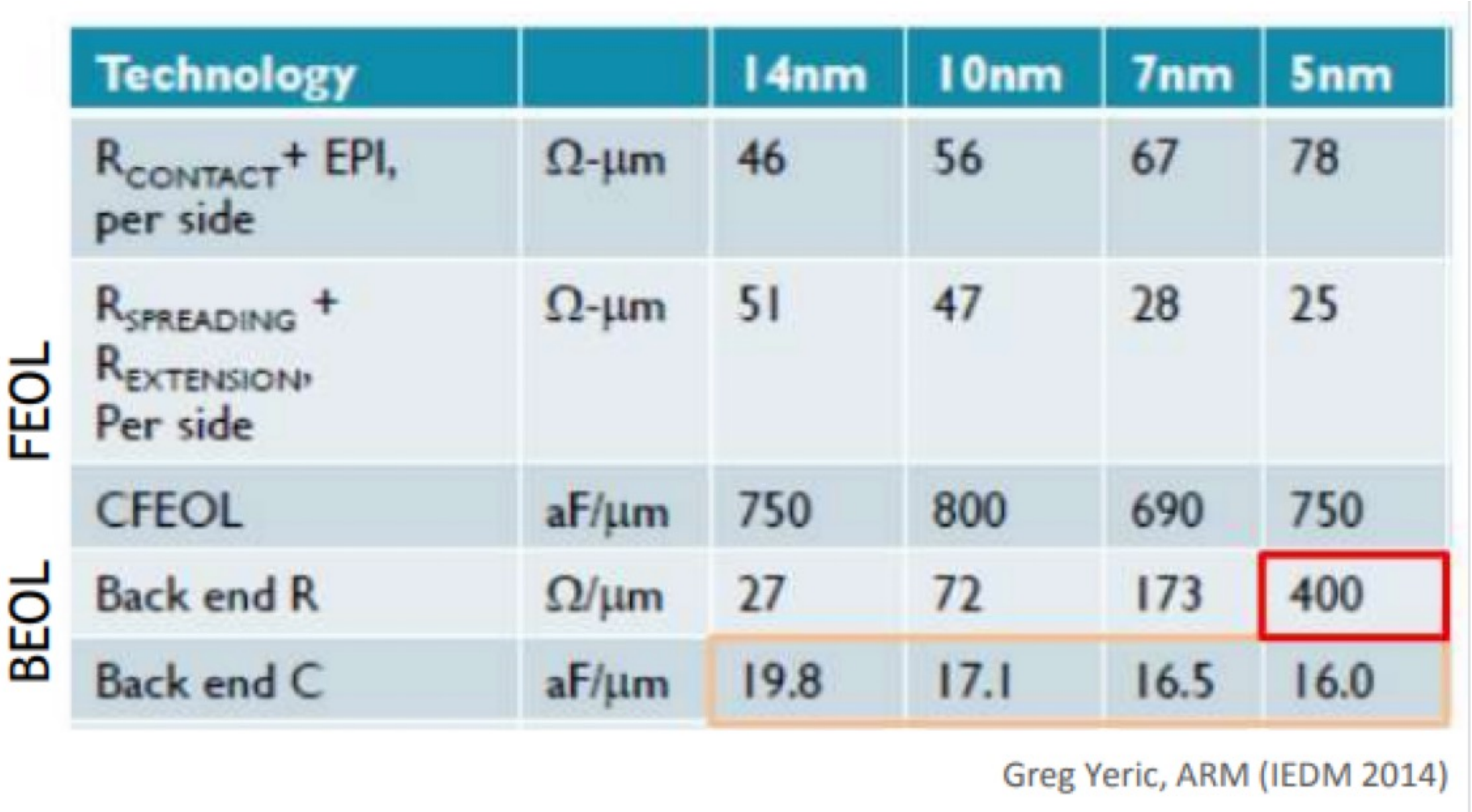
Clock Frequency vs RC



The problem arise from 3 main sectors - BEOL RC delay, thermal consideration and product reliability.

BEOL has become the main bottleneck in CPU performance - once we've passed the 22 nm node. **Basically we have state-of-the-art transistors that do not get enough "juice" because the interconnects are too slow (combination of resistivity and parasitic capacity).**

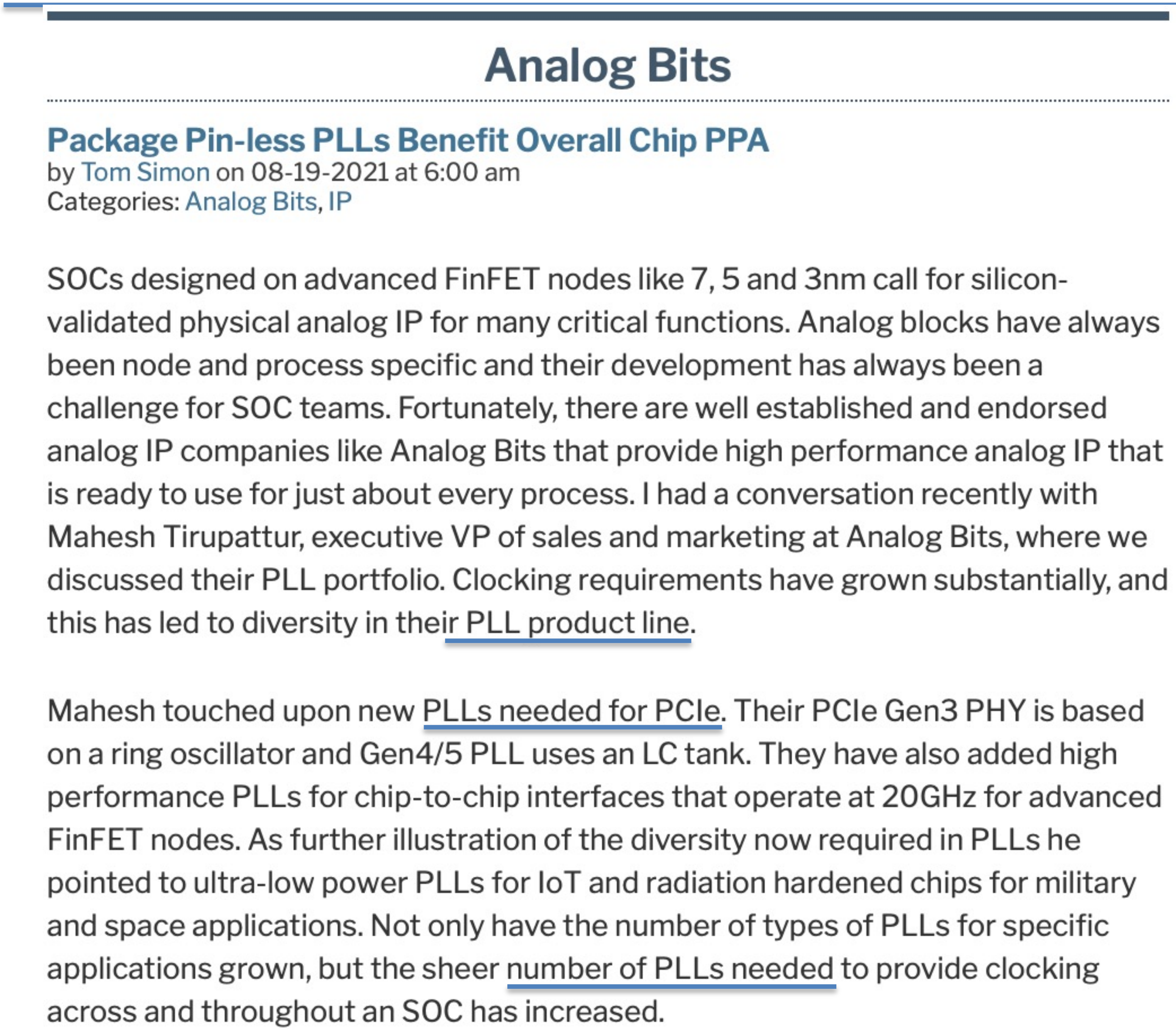
Clock Frequency vs RC



	Technology		14nm	10nm	7nm	5nm
FEOL	$R_{CONTACT}$ + EPI, per side	Ω-µm	46	56	67	78
	$R_{SPREADING}$ + $R_{EXTENSION}$, Per side	Ω-µm	51	47	28	25
	CFEOL	aF/µm	750	800	690	750
BEOL	Back end R	Ω/µm	27	72	173	400
	Back end C	aF/µm	19.8	17.1	16.5	16.0

Greg Yeric, ARM (IEDM 2014)

Clocks: PLL



Analog Bits

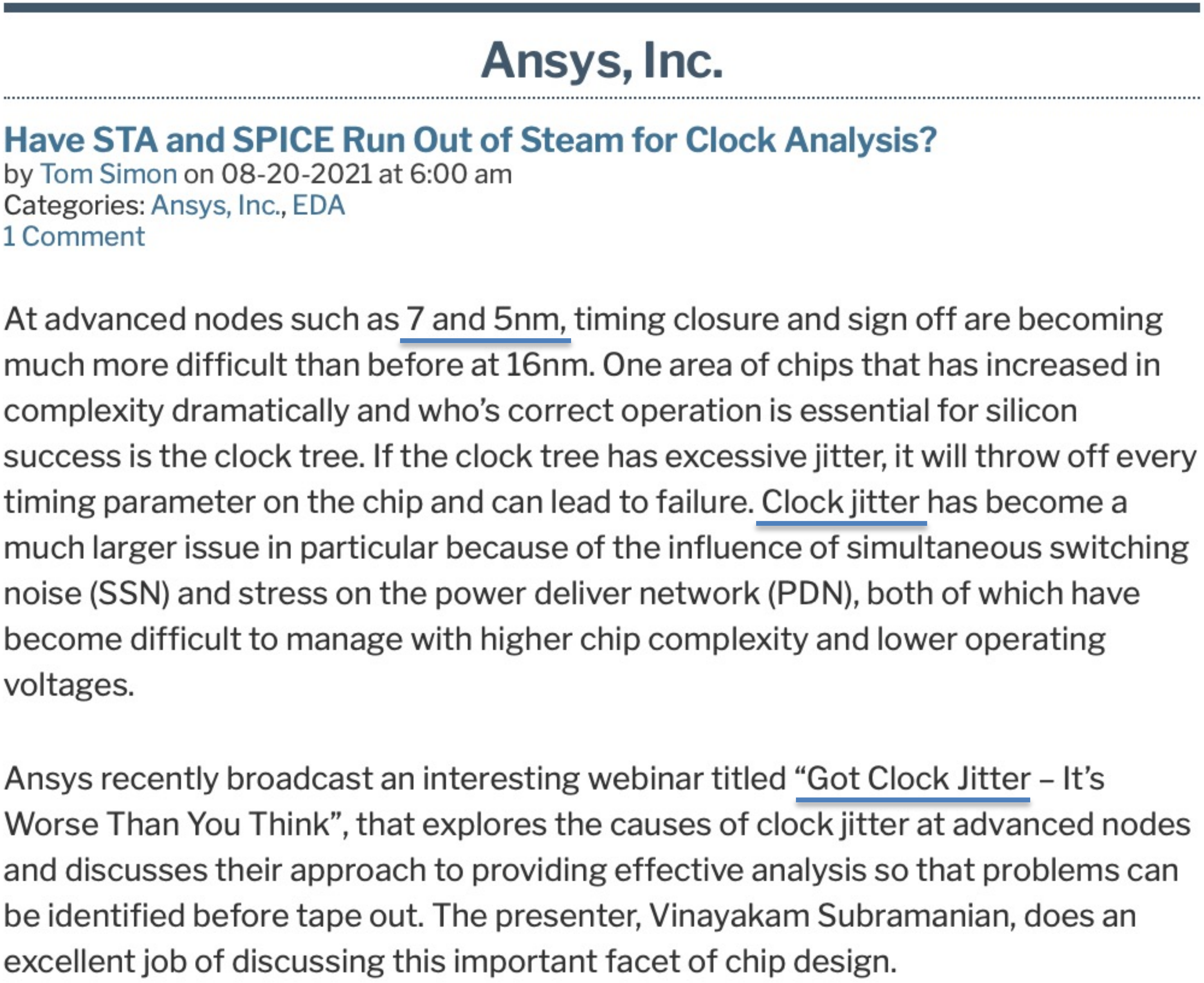
Package Pin-less PLLs Benefit Overall Chip PPA

by Tom Simon on 08-19-2021 at 6:00 am
Categories: Analog Bits, IP

SOCs designed on advanced FinFET nodes like 7, 5 and 3nm call for silicon-validated physical analog IP for many critical functions. Analog blocks have always been node and process specific and their development has always been a challenge for SOC teams. Fortunately, there are well established and endorsed analog IP companies like Analog Bits that provide high performance analog IP that is ready to use for just about every process. I had a conversation recently with Mahesh Tirupattur, executive VP of sales and marketing at Analog Bits, where we discussed their PLL portfolio. Clocking requirements have grown substantially, and this has led to diversity in their PLL product line.

Mahesh touched upon new PLLs needed for PCIe. Their PCIe Gen3 PHY is based on a ring oscillator and Gen4/5 PLL uses an LC tank. They have also added high performance PLLs for chip-to-chip interfaces that operate at 20GHz for advanced FinFET nodes. As further illustration of the diversity now required in PLLs he pointed to ultra-low power PLLs for IoT and radiation hardened chips for military and space applications. Not only have the number of types of PLLs for specific applications grown, but the sheer number of PLLs needed to provide clocking across and throughout an SOC has increased.

Clocks: Jitter



Ansys, Inc.

Have STA and SPICE Run Out of Steam for Clock Analysis?

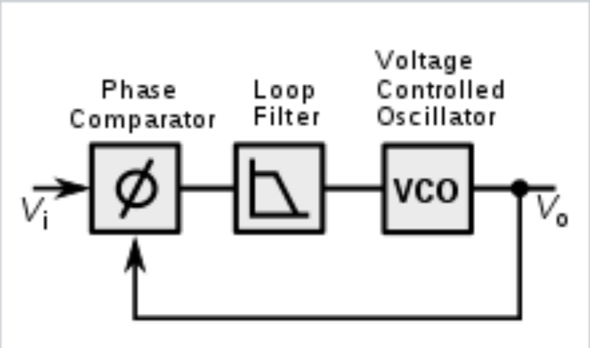
by Tom Simon on 08-20-2021 at 6:00 am
Categories: Ansys, Inc., EDA
1 Comment

At advanced nodes such as 7 and 5nm, timing closure and sign off are becoming much more difficult than before at 16nm. One area of chips that has increased in complexity dramatically and who's correct operation is essential for silicon success is the clock tree. If the clock tree has excessive jitter, it will throw off every timing parameter on the chip and can lead to failure. Clock jitter has become a much larger issue in particular because of the influence of simultaneous switching noise (SSN) and stress on the power deliver network (PDN), both of which have become difficult to manage with higher chip complexity and lower operating voltages.

Ansys recently broadcast an interesting webinar titled "Got Clock Jitter – It's Worse Than You Think", that explores the causes of clock jitter at advanced nodes and discusses their approach to providing effective analysis so that problems can be identified before tape out. The presenter, Vinayakam Subramanian, does an excellent job of discussing this important facet of chip design.

Stable Clocks

- ❖ PLL (Phase-locked Loop)



Simplest analog phase locked loop

F1

Frequency stability metrics

- ❖ Jitter (ppm)
- ❖ Wander (ppm/day)
  - ➢ **Stratum 1/2/3/4**

NIST-F1 is a cesium fountain clock, a type of atomic clock, in the National Institute of Standards and Technology (NIST) in Boulder, Colorado, and serves as the United States' primary time and frequency standard. The clock took less than four years to test and build, and was developed

F2

NIST physicists Steve Jefferts (foreground) and Tom Heavner with the NIST-F2 cesium fountain atomic clock, a civilian time standard for the United States.

NIST-F2

From Wikipedia, the free encyclopedia

- ❖ US Atomic clock (NIST-F1/2)

NIST-F2 is a caesium fountain atomic clock that, along with NIST-F1, serves as the United States' primary time and frequency standard.[1] NIST-F2 was brought online on 3 April 2014.[1][2]

Accuracy [edit]

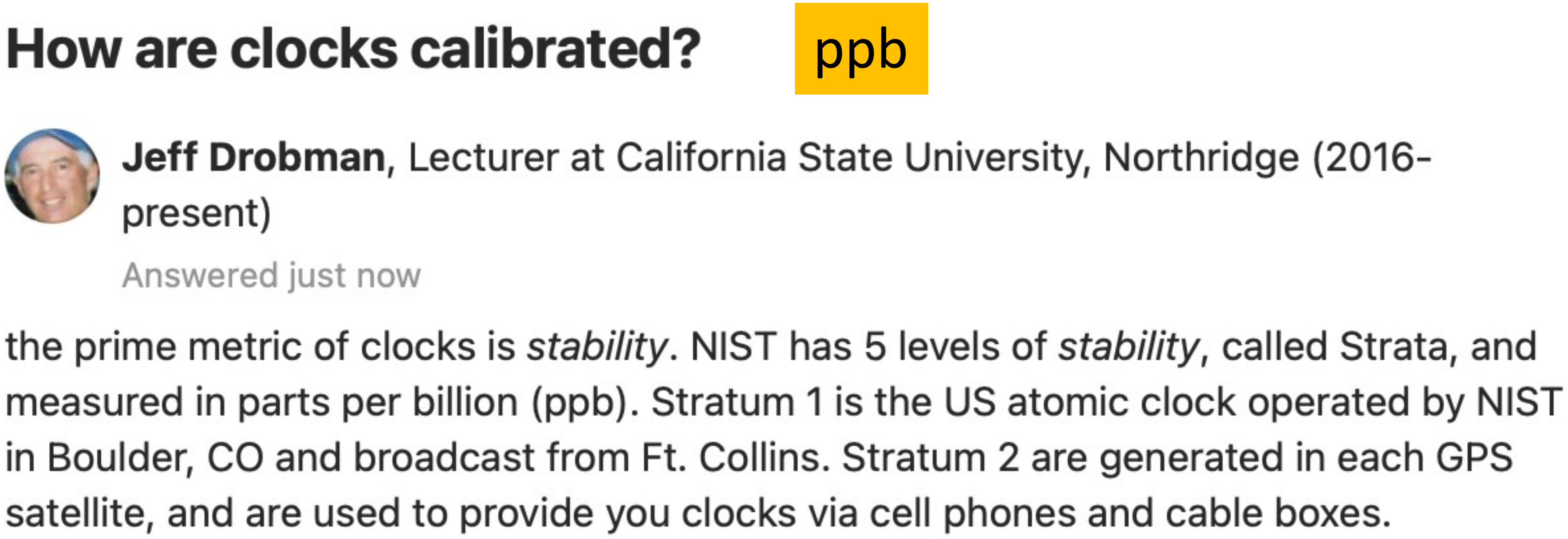
NIST-F1, a caesium fountain atomic clock used since 1999, has a fractional inaccuracy ($\delta f / f$) of less than 5×10^{-16}.

The planned performance of NIST-F2 is $\delta f / f < 1 \times 10^{-16}$.[3] At this planned performance level the NIST-F2 clock will not lose a second in at least 300 million years.[4]

300,000,000 years!

Clock Stability

ppb



How are clocks calibrated?

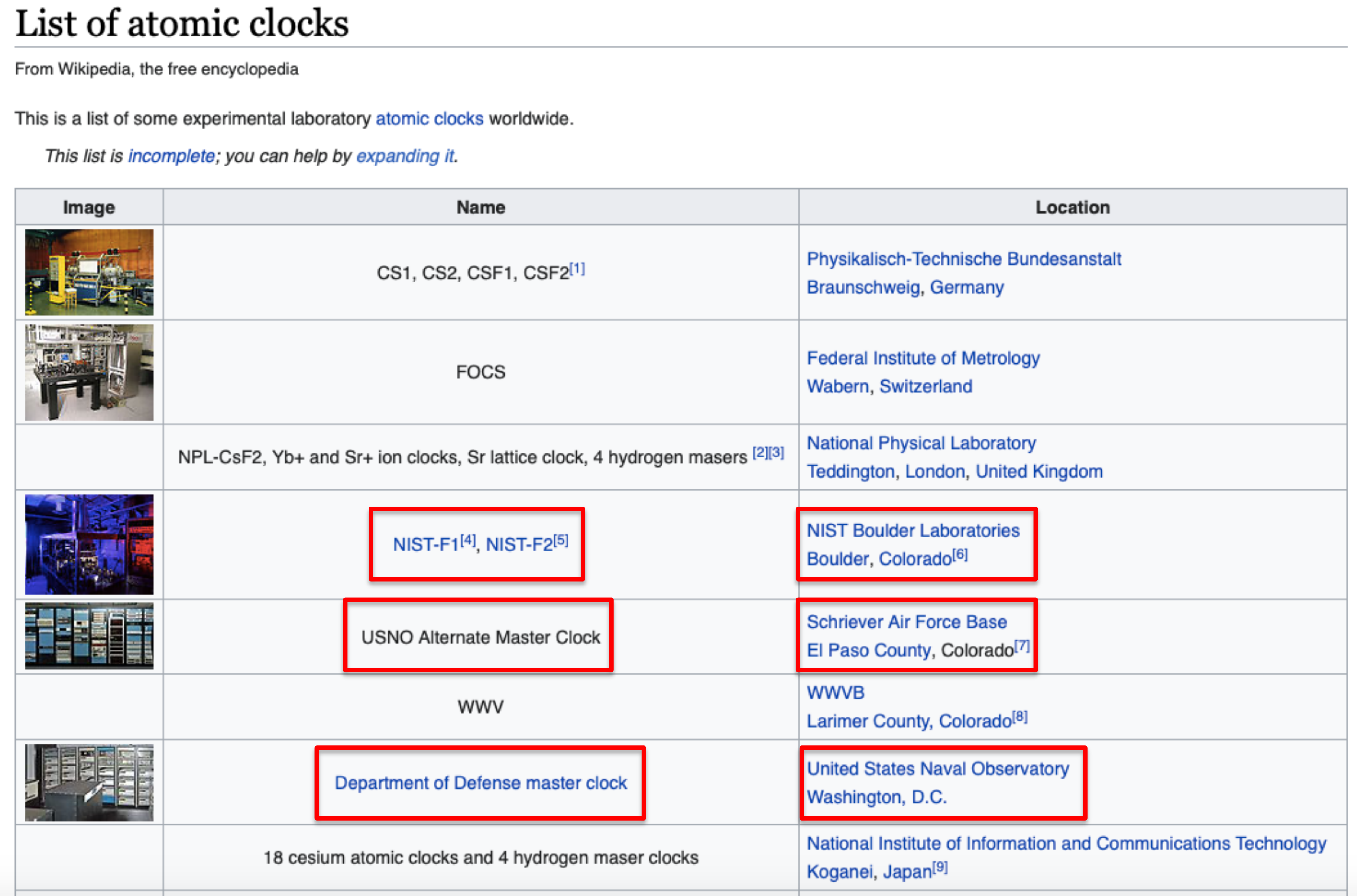
Jeff Drobman, Lecturer at California State University, Northridge (2016-present)

Answered just now

the prime metric of clocks is *stability*. NIST has 5 levels of *stability*, called Strata, and measured in parts per billion (ppb). Stratum 1 is the US atomic clock operated by NIST in Boulder, CO and broadcast from Ft. Collins. Stratum 2 are generated in each GPS satellite, and are used to provide you clocks via cell phones and cable boxes.

- Stratum 1 → Atomic
- Stratum 2 → GPS → phones, cable boxes
- Stratum 3 → Telco Stations

US Atomic Clocks +WW



List of atomic clocks

From Wikipedia, the free encyclopedia

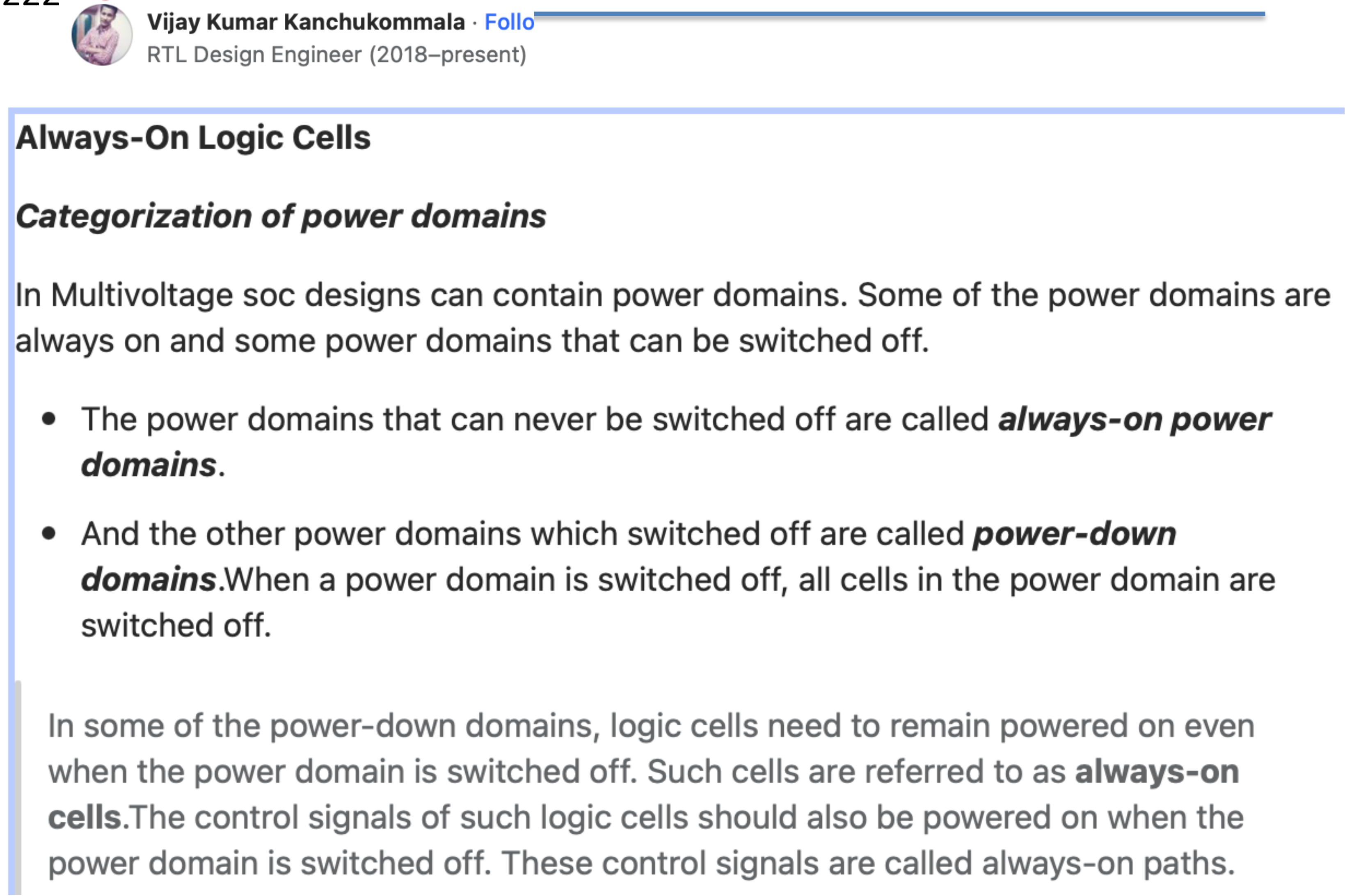
This is a list of some experimental laboratory atomic clocks worldwide.

This list is incomplete; you can help by expanding it.

Image	Name	Location
	CS1, CS2, CSF1, CSF2[1]	Physikalisch-Technische Bundesanstalt Braunschweig, Germany
	FOCS	Federal Institute of Metrology Wabern, Switzerland
	NPL-CsF2, Yb+ and Sr+ ion clocks, Sr lattice clock, 4 hydrogen masers [2][3]	National Physical Laboratory Teddington, London, United Kingdom
	NIST-F1[4], NIST-F2[5]	NIST Boulder Laboratories Boulder, Colorado[6]
	USNO Alternate Master Clock	Schriever Air Force Base El Paso County, Colorado[7]
	WWV	WWVB Larimer County, Colorado[8]
	Department of Defense master clock	United States Naval Observatory Washington, D.C.
	18 cesium atomic clocks and 4 hydrogen maser clocks	National Institute of Information and Communications Technology Koganei, Japan[9]

Clock Domains: On/Off

Quora



Vijay Kumar Kanchukommala · Follo
RTL Design Engineer (2018–present)

Always-On Logic Cells

Categorization of power domains

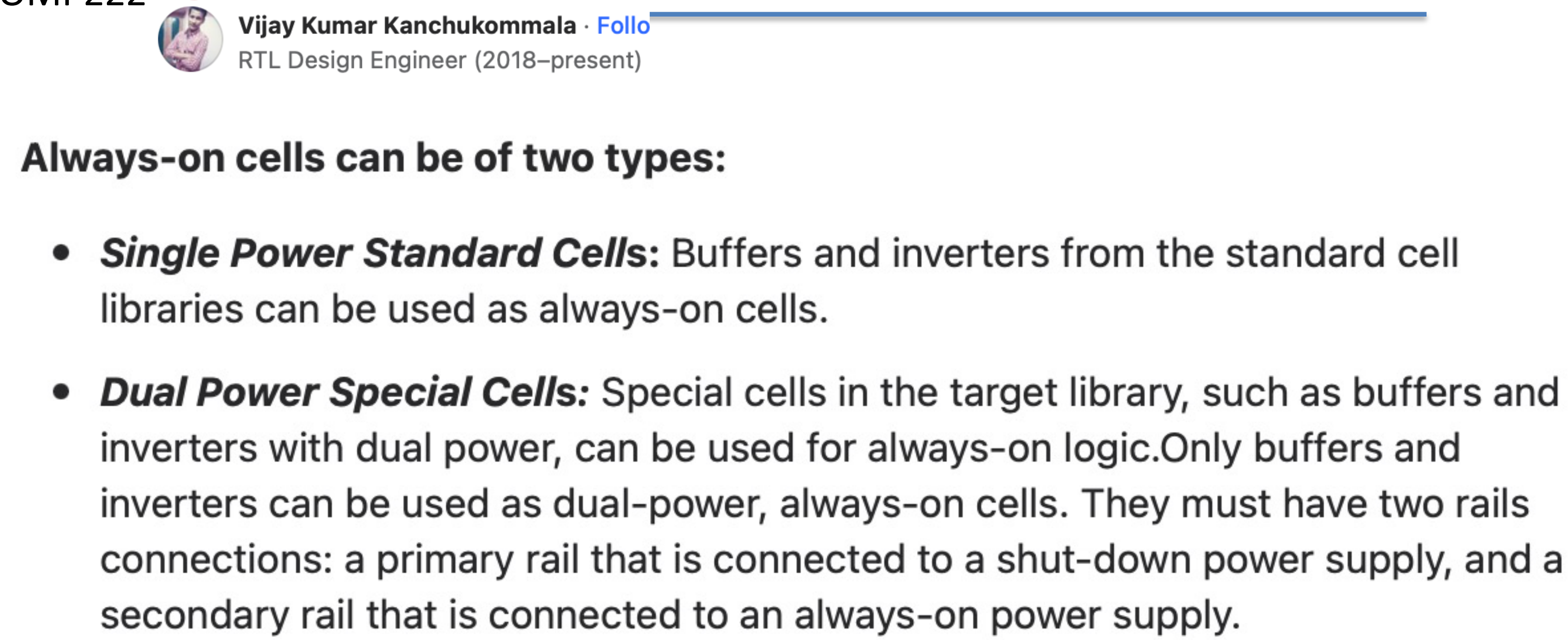
In Multivoltage soc designs can contain power domains. Some of the power domains are always on and some power domains that can be switched off.

- The power domains that can never be switched off are called ***always-on power domains***.
- And the other power domains which switched off are called ***power-down domains***.When a power domain is switched off, all cells in the power domain are switched off.

> In some of the power-down domains, logic cells need to remain powered on even when the power domain is switched off. Such cells are referred to as **always-on cells**.The control signals of such logic cells should also be powered on when the power domain is switched off. These control signals are called always-on paths.

Clock Domains: On/Off

Quora



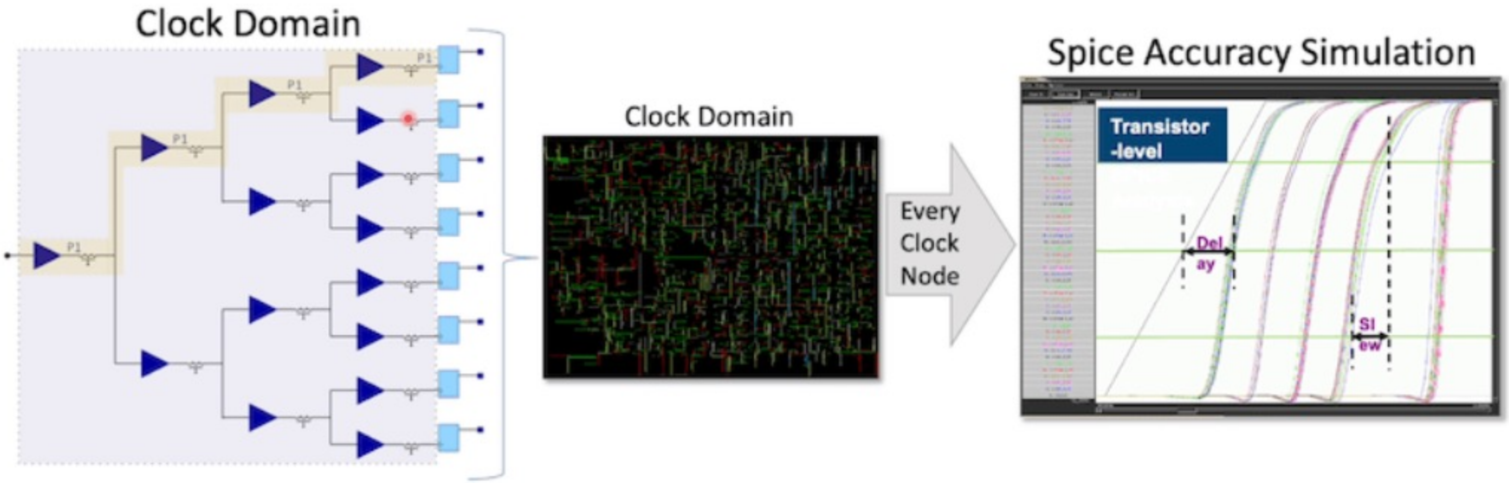
Vijay Kumar Kanchukommala · Follo
RTL Design Engineer (2018–present)

Always-on cells can be of two types:

- ***Single Power Standard Cells*:** Buffers and inverters from the standard cell libraries can be used as always-on cells.
- ***Dual Power Special Cells:*** Special cells in the target library, such as buffers and inverters with dual power, can be used for always-on logic.Only buffers and inverters can be used as dual-power, always-on cells. They must have two rails connections: a primary rail that is connected to a shut-down power supply, and a secondary rail that is connected to an always-on power supply.

Clock Slew

Delving into the first step, the fresh run analyzes the clock domain from the output of a PLL, all the way through to flip-flops or output pads. This clock domain can be quite large in size with millions of devices, and the transistor-level analysis results show us the delay and slew values.

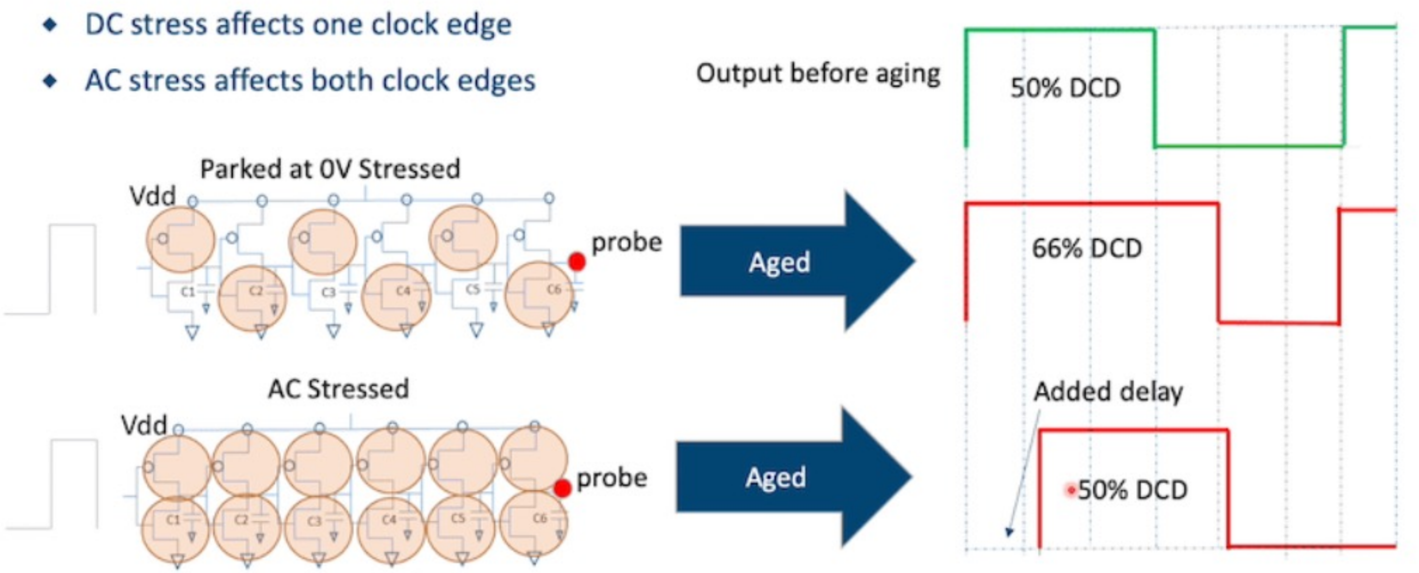


Step 1: Fresh Run

The ClockEdge tool can run clock analysis for a fresh run on a block with 4.5 million gates, 517 million MOSFETs and 3.2 billion devices overnight, by using a distributed SPICE simulation approach. Your clock topology can be implemented as trees, grids and spines.

Clock Aging

The final analysis is Step 3, using the Aged devices. For the case of devices that had a parked clock value, then only one edge of clock will be affected during aging analysis, while devices with clock toggling will have both edges affected during aging analysis. So the Duty Cycle Delay (DCD) shape will depend on your circuit topology.



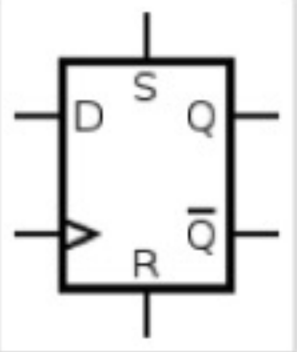
Step 3, Aged Simulation

With ClockEdge a designer can perform what-if stress analysis, comparing the impact of a clock parked at 0, parked at 1, toggling, or even a combination of parked and toggling.

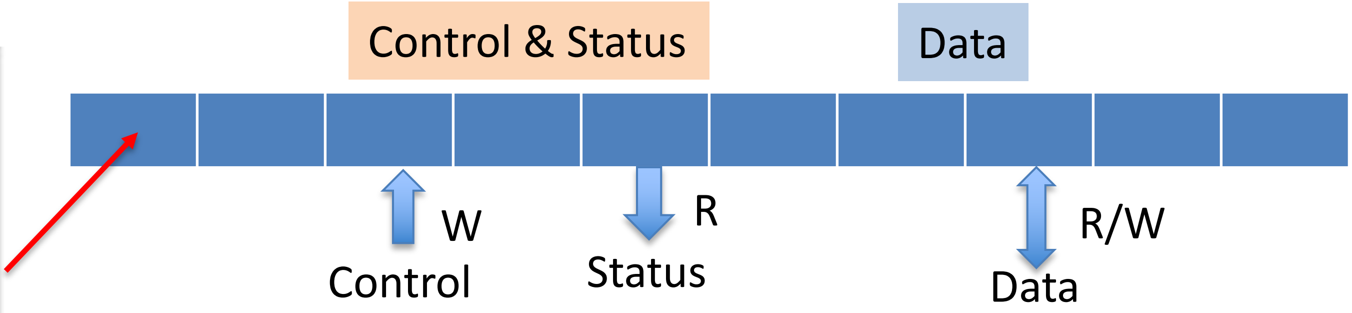
Section

Processor State (PSW)

Registers



D flip-flop symbol



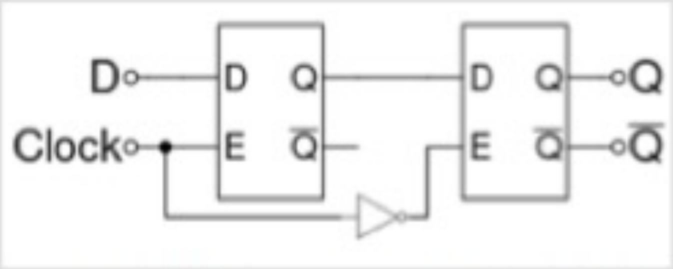
Control & Status

Data

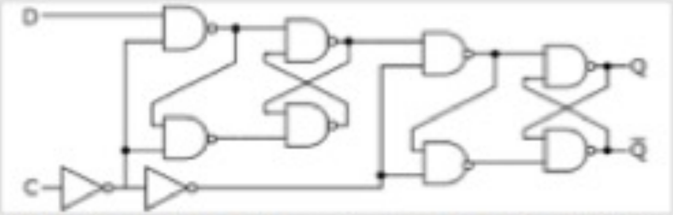
W
Control

R
Status

R/W
Data



A master–slave D flip-flop. It responds on the falling edge of the *enable* input (usually a clock)



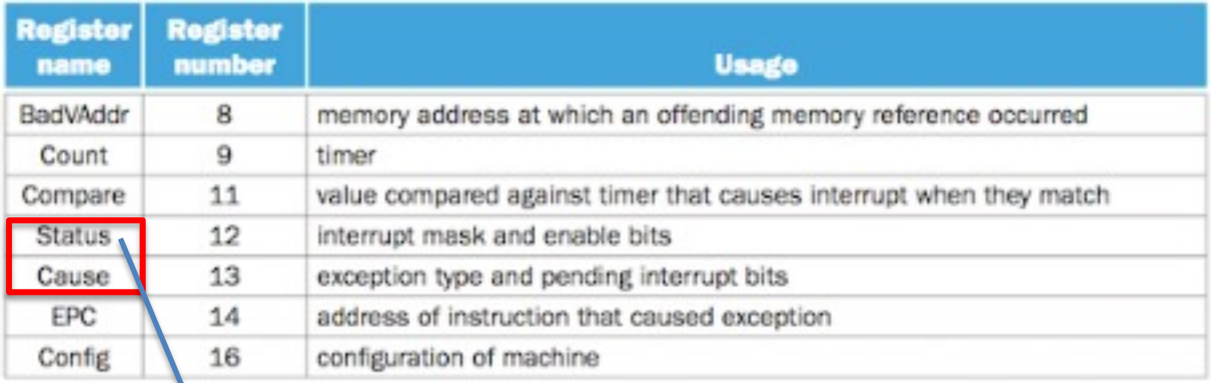
An implementation of a master–slave D flip-flop that is triggered on the rising edge of the clock

Control CP0

7.7 Exceptions and interrupts

Hennessy & Patterson

Figure 7.7.1: Coprocessor 0 registers.



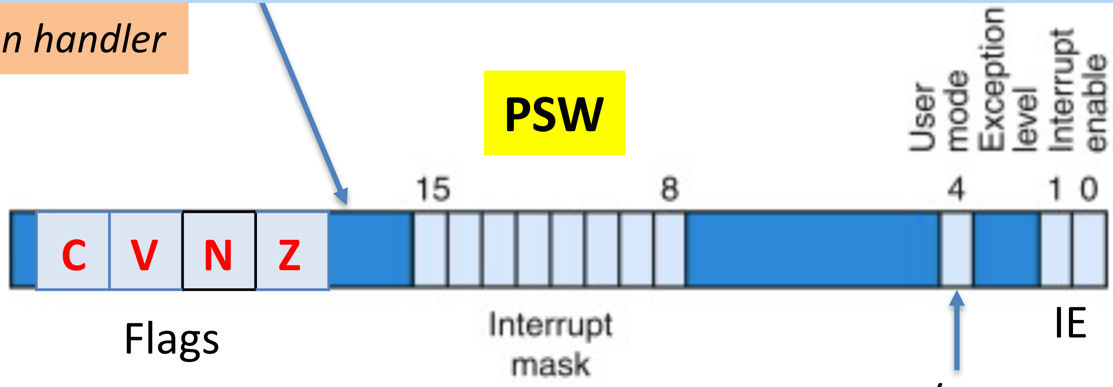
Register name	Register number	Usage
BadVAddr	8	memory address at which an offending memory reference occurred
Count	9	timer
Compare	11	value compared against timer that causes interrupt when they match
Status	12	interrupt mask and enable bits
Cause	13	exception type and pending interrupt bits
EPC	14	address of instruction that caused exception
Config	16	configuration of machine

Figure 7.7.2: The status register (COD Figure

Interrupt handler: A piece of code that is run as a result of an exception or an interrupt.

Exception handler

PSW



Flags

Interrupts(8)

IE

User/system

MIPS: Exception Cause

Hennessy & Patterson

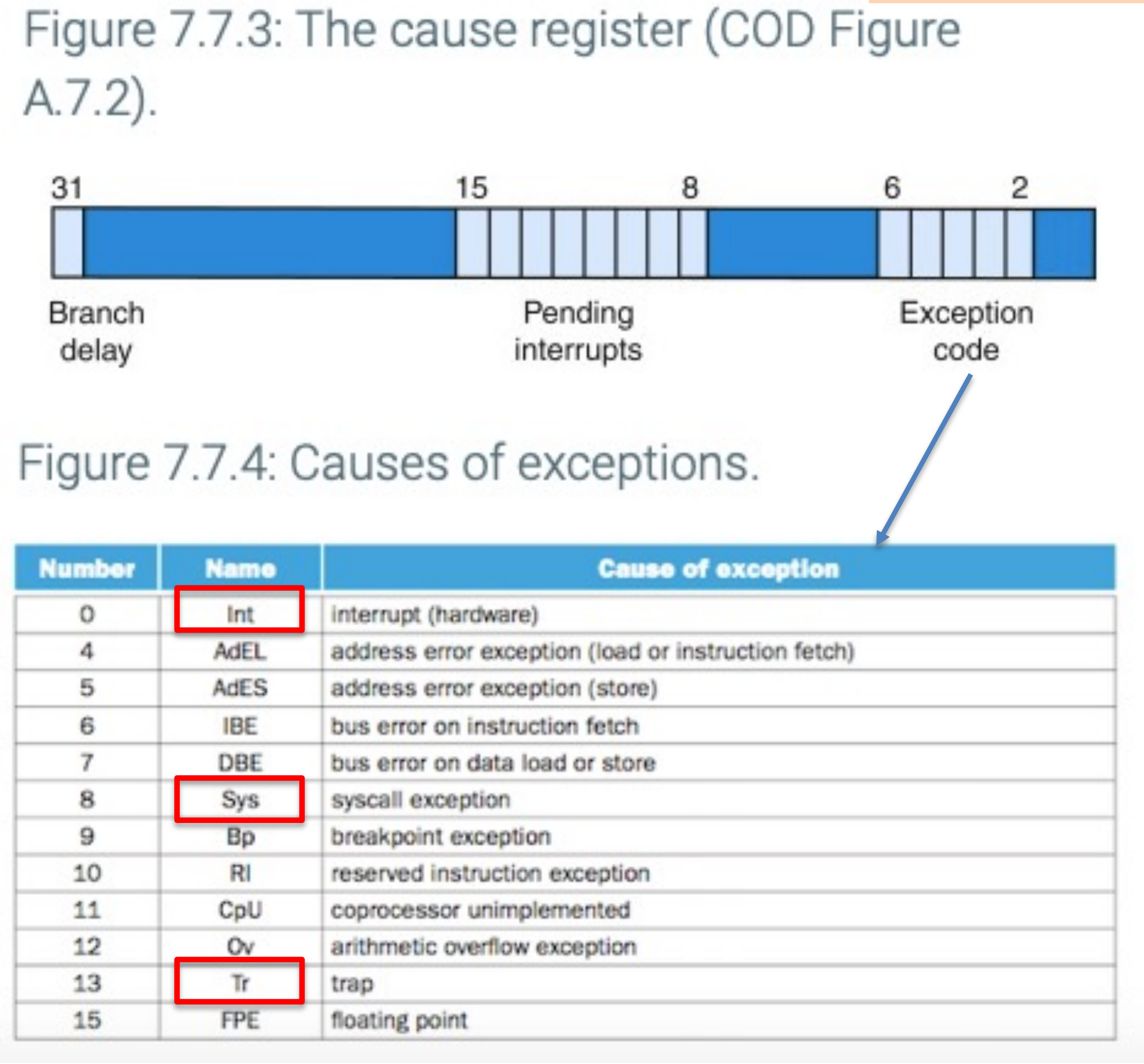


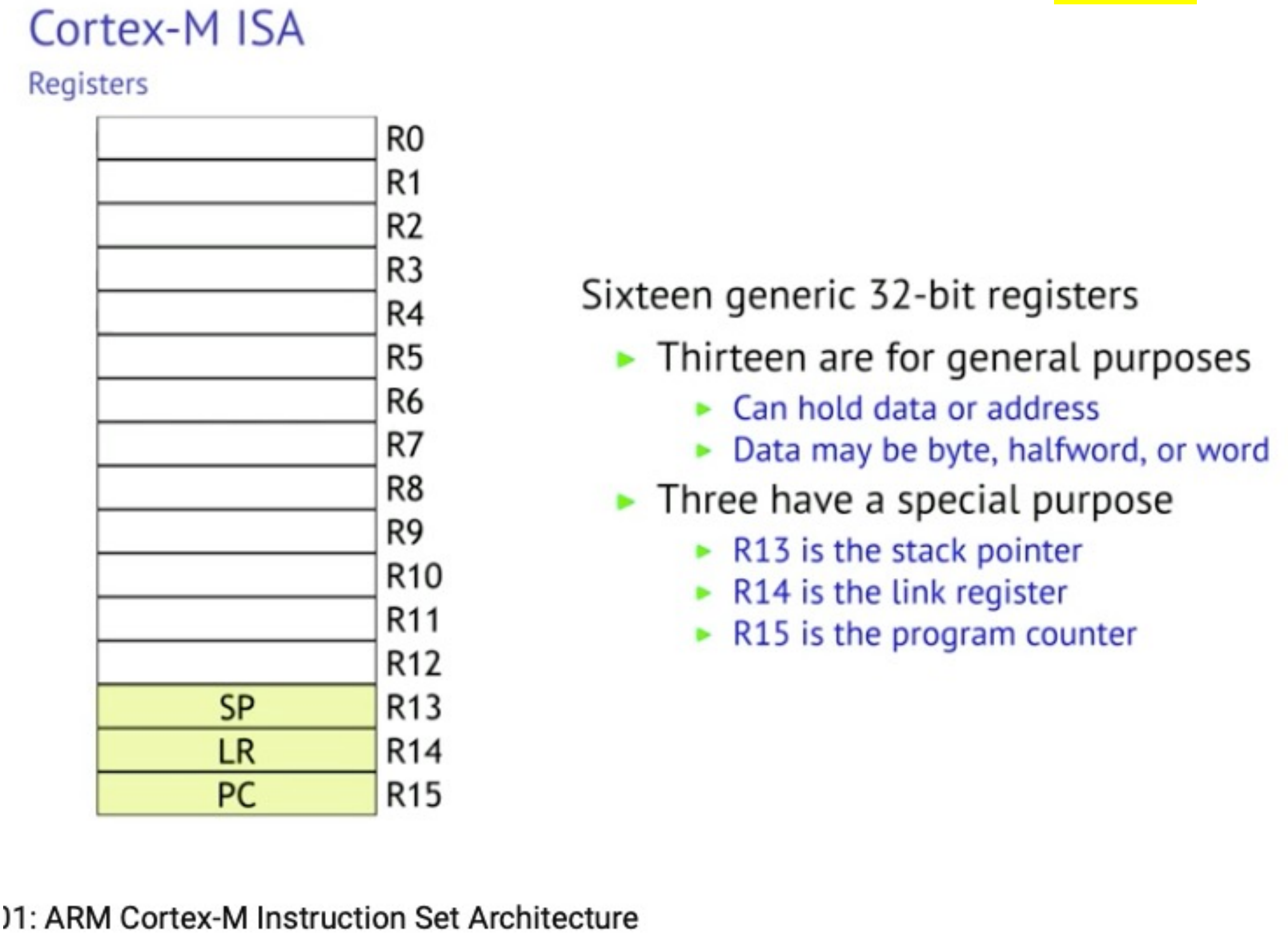
Figure 7.7.3: The cause register (COD Figure A.7.2).

Figure 7.7.4: Causes of exceptions.

Number	Name	Cause of exception
0	Int	interrupt (hardware)
4	AdEL	address error exception (load or instruction fetch)
5	AdES	address error exception (store)
6	IBE	bus error on instruction fetch
7	DBE	bus error on data load or store
8	Sys	syscall exception
9	Bp	breakpoint exception
10	RI	reserved instruction exception
11	CpU	coprocessor unimplemented
12	Ov	arithmetic overflow exception
13	Tr	trap
15	FPE	floating point

ARM

ARMv**7**



Cortex-M ISA

Registers

	R0
	R1
	R2
	R3
	R4
	R5
	R6
	R7
	R8
	R9
	R10
	R11
	R12
SP	R13
LR	R14
PC	R15

Sixteen generic 32-bit registers

- Thirteen are for general purposes
  - Can hold data or address
  - Data may be byte, halfword, or word
- Three have a special purpose
  - R13 is the stack pointer
  - R14 is the link register
  - R15 is the program counter

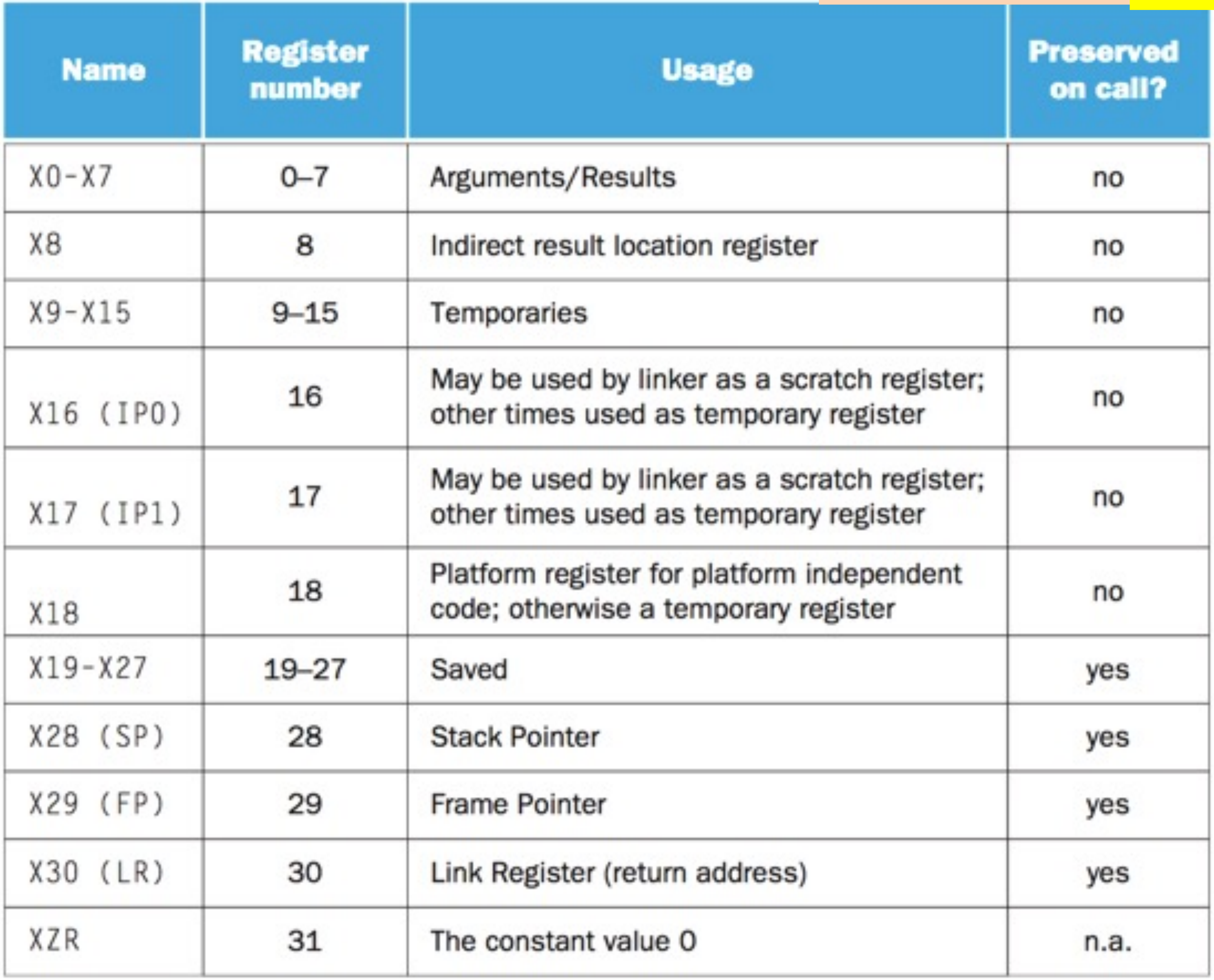
)1: ARM Cortex-M Instruction Set Architecture

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP222

ARM Registers

DR JEFF SOFTWARE
INDIE APP DEVELOPER
© Jeff Drobman
2020-23

Hennessy & Patterson ARMv8



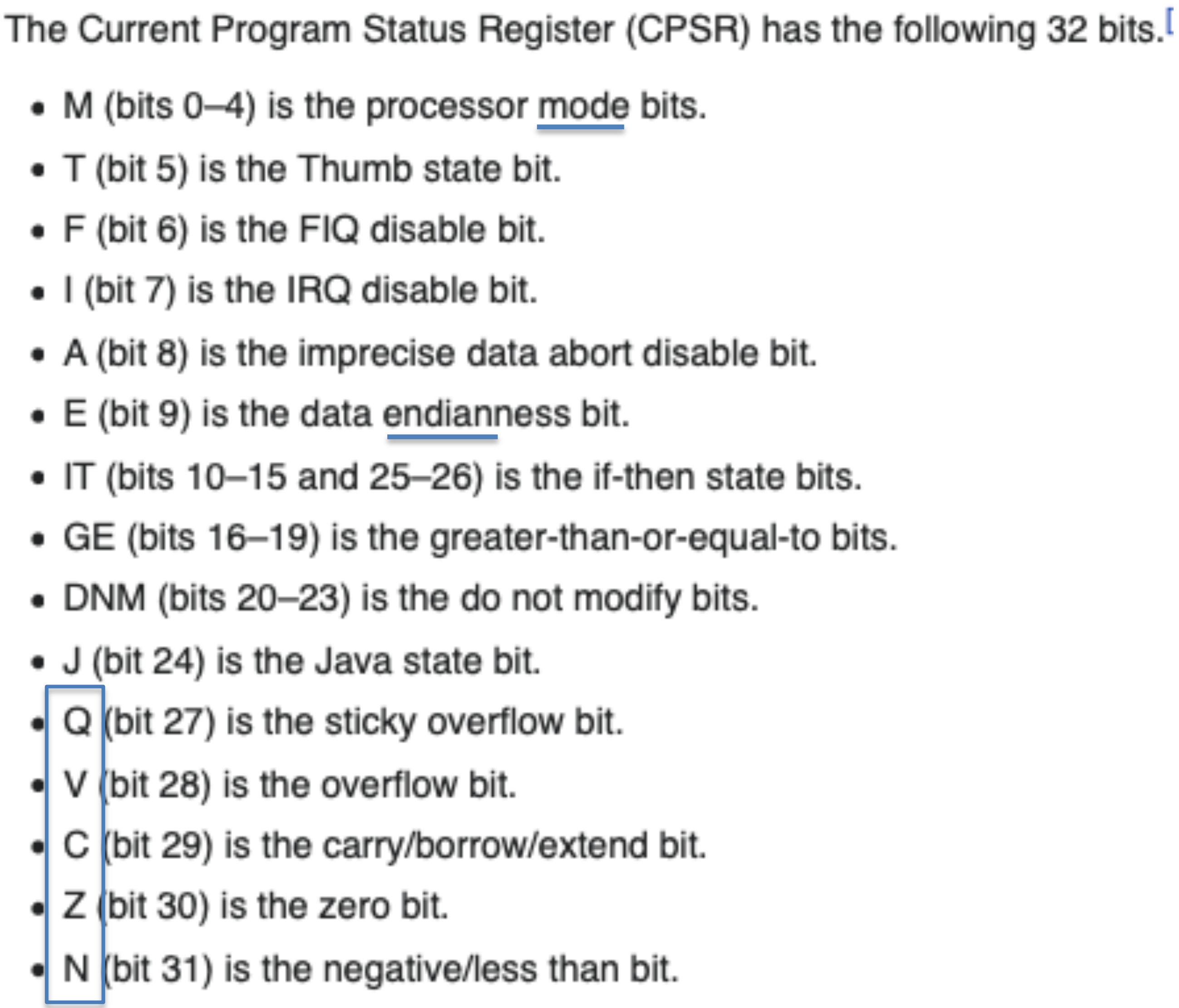
Name	Register number	Usage	Preserved on call?
X0-X7	0–7	Arguments/Results	no
X8	8	Indirect result location register	no
X9-X15	9–15	Temporaries	no
X16 (IP0)	16	May be used by linker as a scratch register; other times used as temporary register	no
X17 (IP1)	17	May be used by linker as a scratch register; other times used as temporary register	no
X18	18	Platform register for platform independent code; otherwise a temporary register	no
X19-X27	19–27	Saved	yes
X28 (SP)	28	Stack Pointer	yes
X29 (FP)	29	Frame Pointer	yes
X30 (LR)	30	Link Register (return address)	yes
XZR	31	The constant value 0	n.a.

COMP222

ARM: Status & Control

© Jeff Drobman
2020-23

CPSR **PSW** ARMv**7**



The Current Program Status Register (CPSR) has the following 32 bits.[

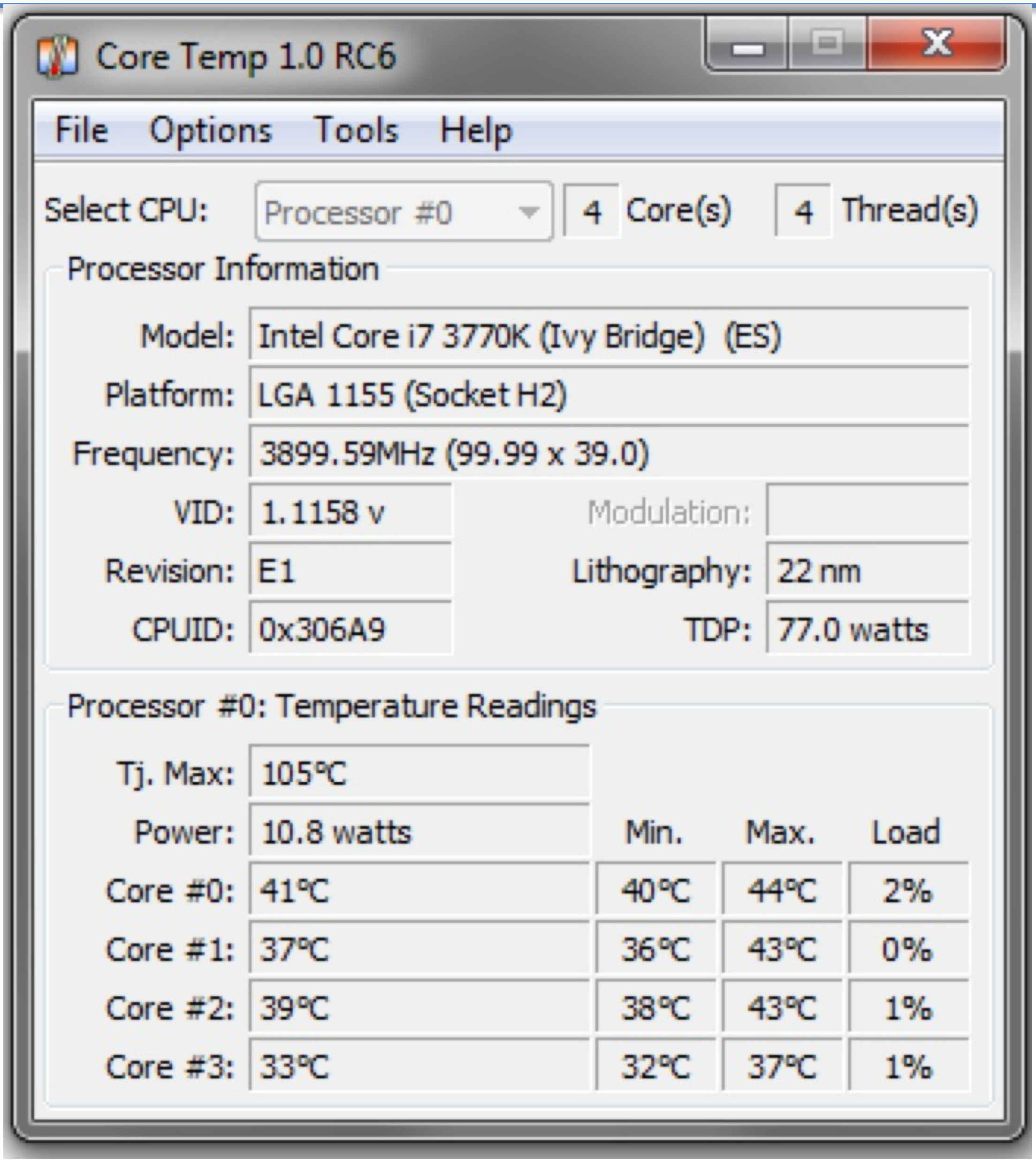
- M (bits 0–4) is the processor mode bits.
- T (bit 5) is the Thumb state bit.
- F (bit 6) is the FIQ disable bit.
- I (bit 7) is the IRQ disable bit.
- A (bit 8) is the imprecise data abort disable bit.
- E (bit 9) is the data endianness bit.
- IT (bits 10–15 and 25–26) is the if-then state bits.
- GE (bits 16–19) is the greater-than-or-equal-to bits.
- DNM (bits 20–23) is the do not modify bits.
- J (bit 24) is the Java state bit.
- Q (bit 27) is the sticky overflow bit.
- V (bit 28) is the overflow bit.
- C (bit 29) is the carry/borrow/extend bit.
- Z (bit 30) is the zero bit.
- N (bit 31) is the negative/less than bit.

Flags

Section

Power Consumption

Power



Select CPU: Processor #0 | 4 Core(s) | 4 Thread(s)

Processor Information

Model: Intel Core i7 3770K (Ivy Bridge) (ES)

Platform: LGA 1155 (Socket H2)

Frequency: 3899.59MHz (99.99 x 39.0)

VID: 1.1158 v

Modulation:

Revision: E1

Lithography: 22 nm

CPUID: 0x306A9

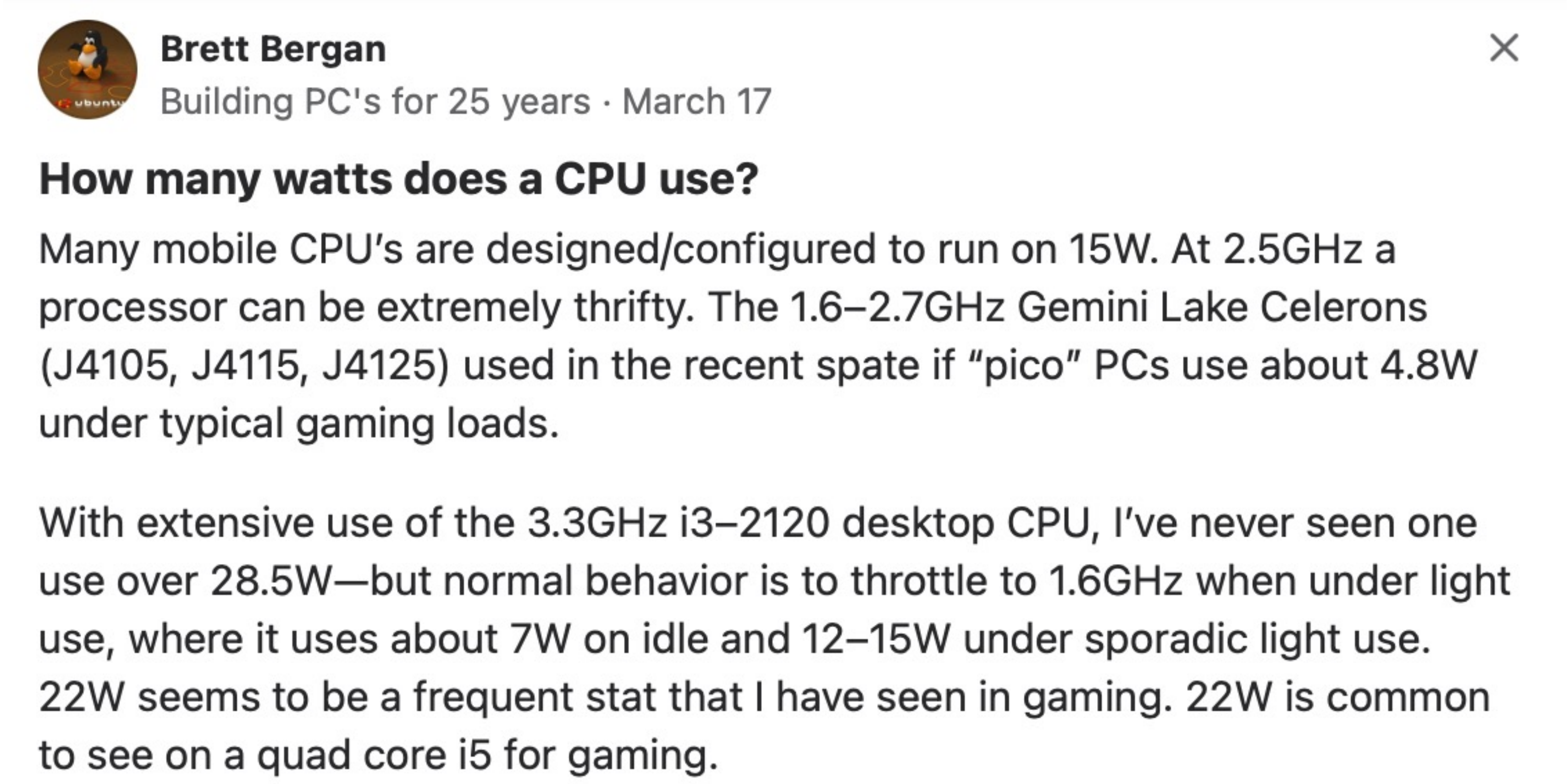
TDP: 77.0 watts

Processor #0: Temperature Readings

Tj. Max: 105°C

Power:	10.8 watts	Min.	Max.	Load
Core #0:	41°C	40°C	44°C	2%
Core #1:	37°C	36°C	43°C	0%
Core #2:	39°C	38°C	43°C	1%
Core #3:	33°C	32°C	37°C	1%

Power Consumption



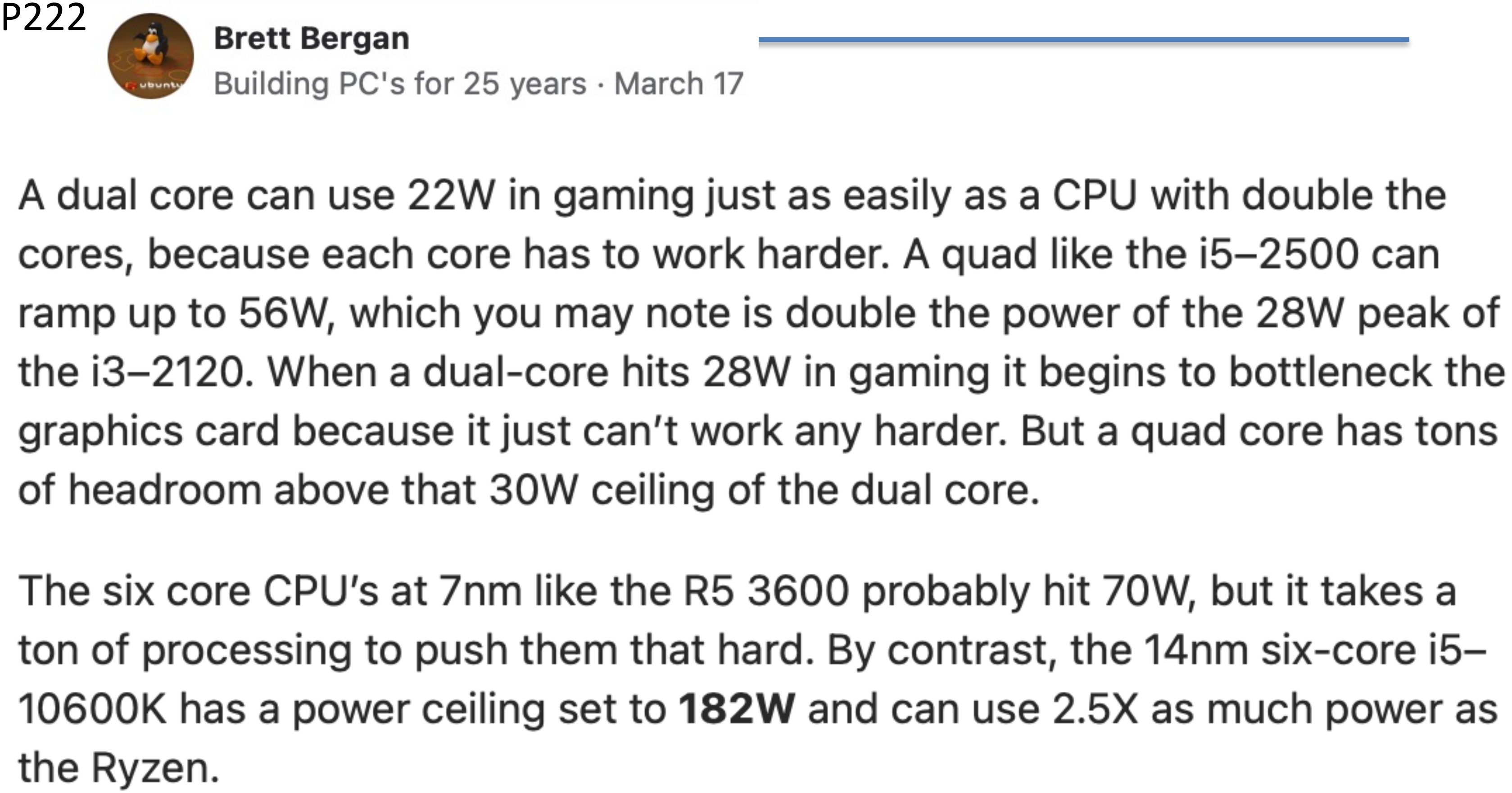
Brett Bergan
Building PC's for 25 years · March 17

How many watts does a CPU use?

Many mobile CPU's are designed/configured to run on 15W. At 2.5GHz a processor can be extremely thrifty. The 1.6–2.7GHz Gemini Lake Celerons (J4105, J4115, J4125) used in the recent spate if "pico" PCs use about 4.8W under typical gaming loads.

With extensive use of the 3.3GHz i3–2120 desktop CPU, I've never seen one use over 28.5W—but normal behavior is to throttle to 1.6GHz when under light use, where it uses about 7W on idle and 12–15W under sporadic light use. 22W seems to be a frequent stat that I have seen in gaming. 22W is common to see on a quad core i5 for gaming.

Power Consumption

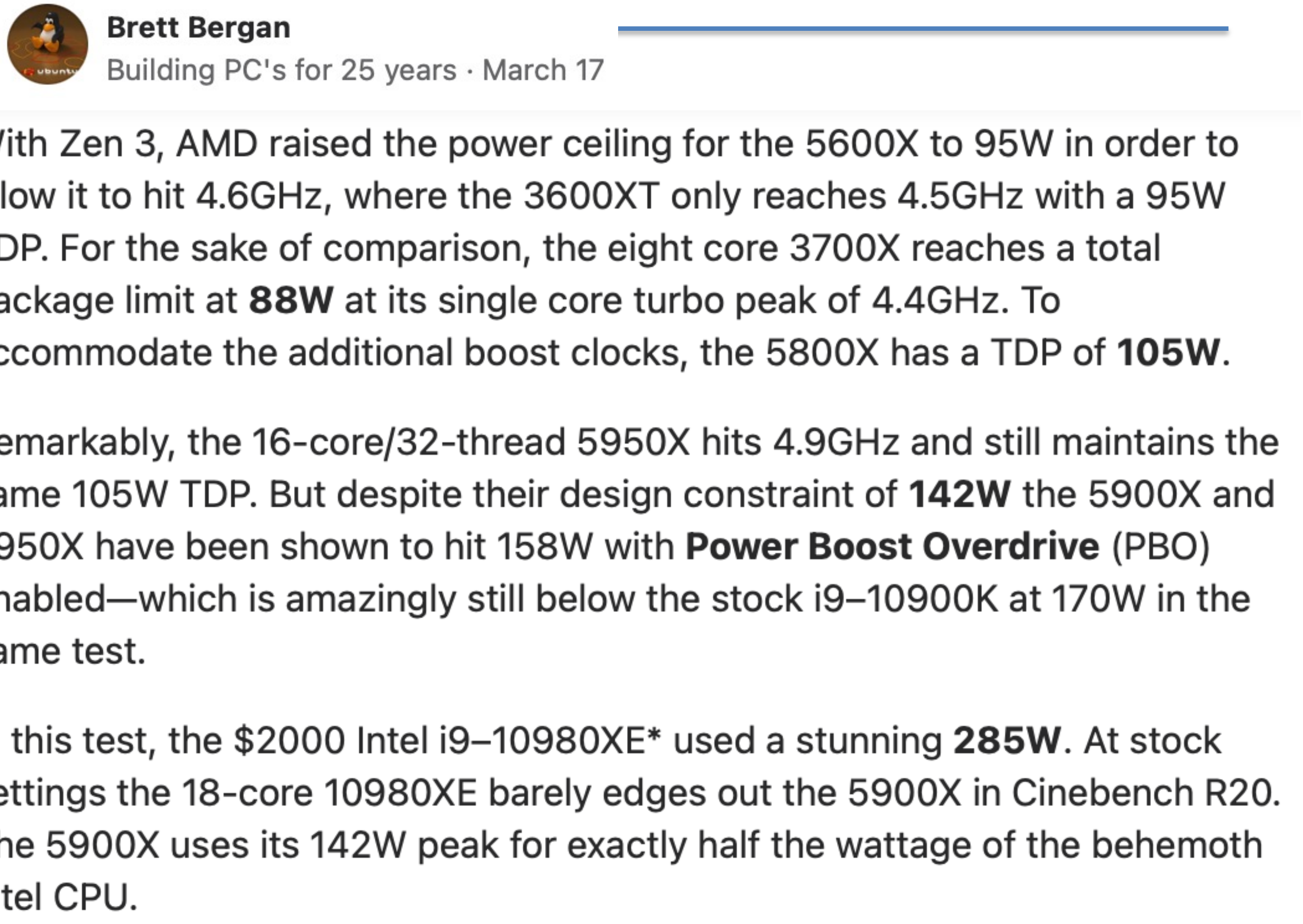


Brett Bergan
Building PC's for 25 years · March 17

A dual core can use 22W in gaming just as easily as a CPU with double the cores, because each core has to work harder. A quad like the i5–2500 can ramp up to 56W, which you may note is double the power of the 28W peak of the i3–2120. When a dual-core hits 28W in gaming it begins to bottleneck the graphics card because it just can't work any harder. But a quad core has tons of headroom above that 30W ceiling of the dual core.

The six core CPU's at 7nm like the R5 3600 probably hit 70W, but it takes a ton of processing to push them that hard. By contrast, the 14nm six-core i5–10600K has a power ceiling set to **182W** and can use 2.5X as much power as the Ryzen.

Power Consumption



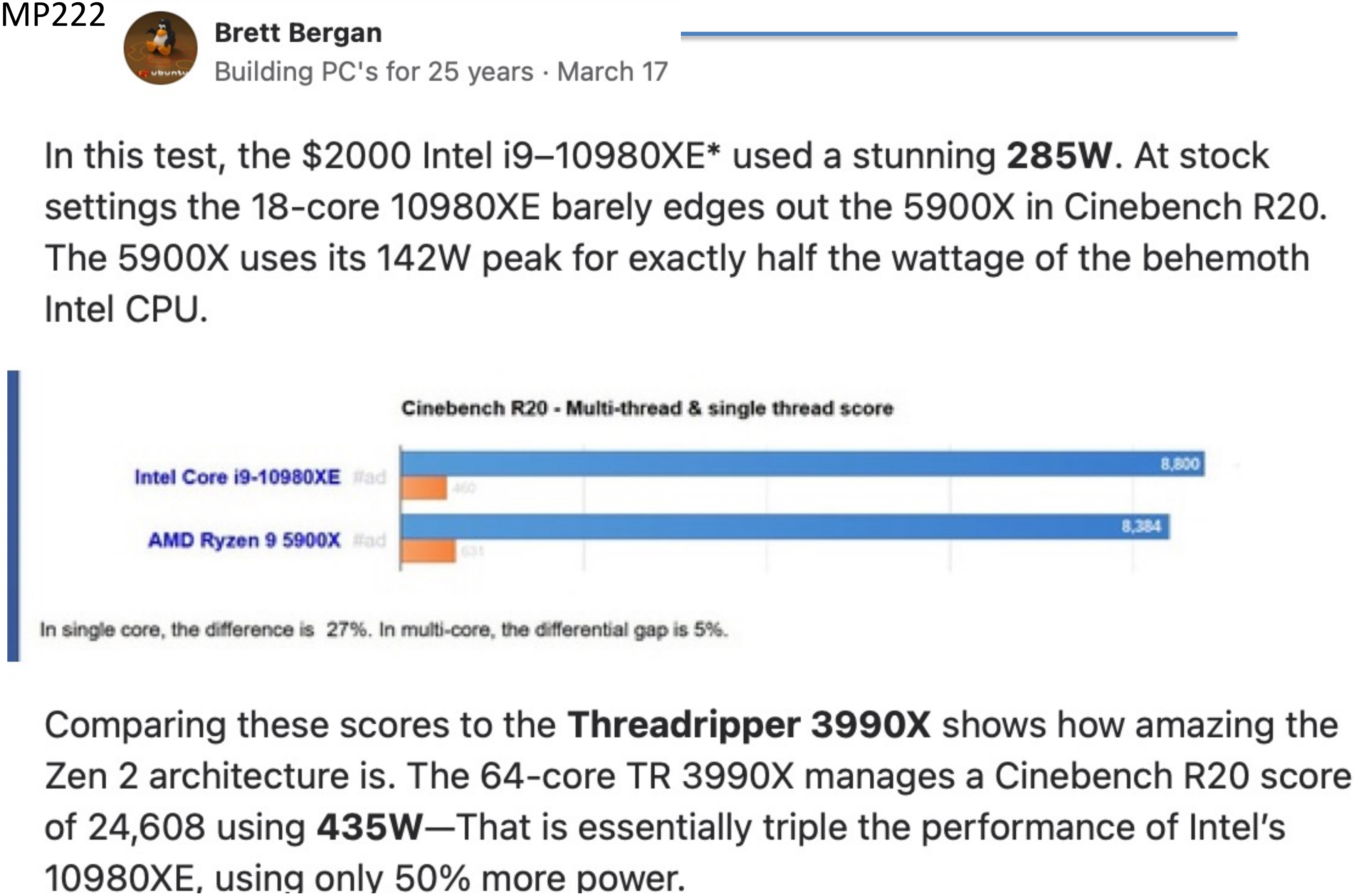
Brett Bergan
Building PC's for 25 years · March 17

With Zen 3, AMD raised the power ceiling for the 5600X to 95W in order to allow it to hit 4.6GHz, where the 3600XT only reaches 4.5GHz with a 95W TDP. For the sake of comparison, the eight core 3700X reaches a total package limit at **88W** at its single core turbo peak of 4.4GHz. To accommodate the additional boost clocks, the 5800X has a TDP of **105W**.

Remarkably, the 16-core/32-thread 5950X hits 4.9GHz and still maintains the same 105W TDP. But despite their design constraint of **142W** the 5900X and 5950X have been shown to hit 158W with **Power Boost Overdrive** (PBO) enabled—which is amazingly still below the stock i9–10900K at 170W in the same test.

In this test, the $2000 Intel i9–10980XE* used a stunning **285W**. At stock settings the 18-core 10980XE barely edges out the 5900X in Cinebench R20. The 5900X uses its 142W peak for exactly half the wattage of the behemoth Intel CPU.

Power Consumption



Brett Bergan

Building PC's for 25 years · March 17

In this test, the $2000 Intel i9–10980XE* used a stunning **285W**. At stock settings the 18-core 10980XE barely edges out the 5900X in Cinebench R20. The 5900X uses its 142W peak for exactly half the wattage of the behemoth Intel CPU.

Cinebench R20 - Multi-thread & single thread score

In single core, the difference is 27%. In multi-core, the differential gap is 5%.

Comparing these scores to the **Threadripper 3990X** shows how amazing the Zen 2 architecture is. The 64-core TR 3990X manages a Cinebench R20 score of 24,608 using **435W**—That is essentially triple the performance of Intel's 10980XE, using only 50% more power.