Computer Architecture
컴퓨터 구조

Lecture 1 : Introduction

미래관 718호
임은진

Assessment

- **Quiz + Homework** (30%)
- 출석 (10%)
  - 9/12 부터 반영
  - 전자 출석 (학기 중 11.25 시간(7.5회) 이상 출석이면 감점 없음)
- 중간 고사 (30%)
- 기말 고사 (30%)
  - 중간 고사 대면 10/24 화 오후 6:00-7:00
  - 기말 고사 대면 12/12 화 오후 6:00-7:00

http://ecampus.kookmin.ac.kr

- 강의 슬라이드 다운로드
- 공지 사항
- QnA

교과목 개요/교육목표

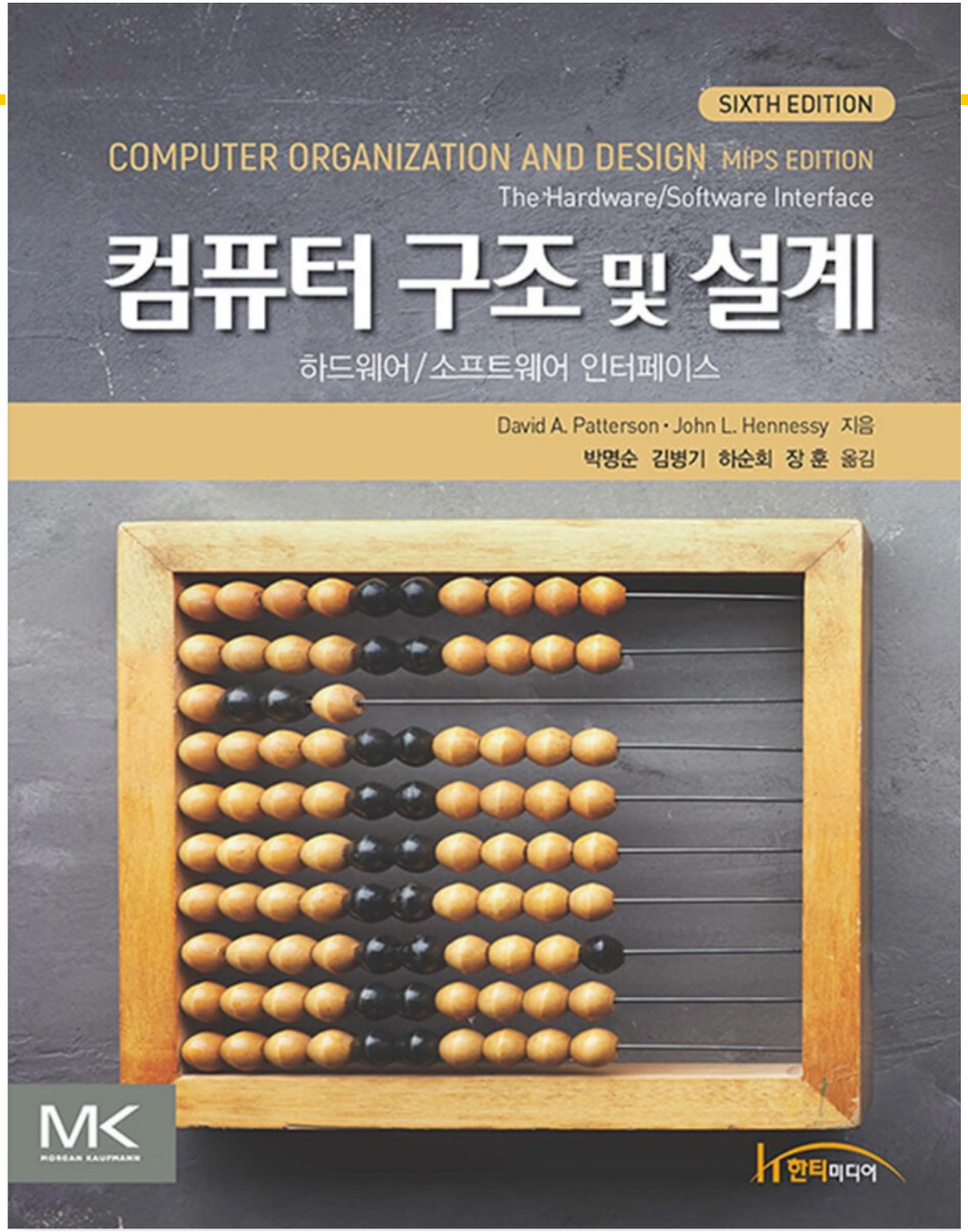
- 컴퓨터의 작동 원리를 디지털 회로 수준에서 이해한다.

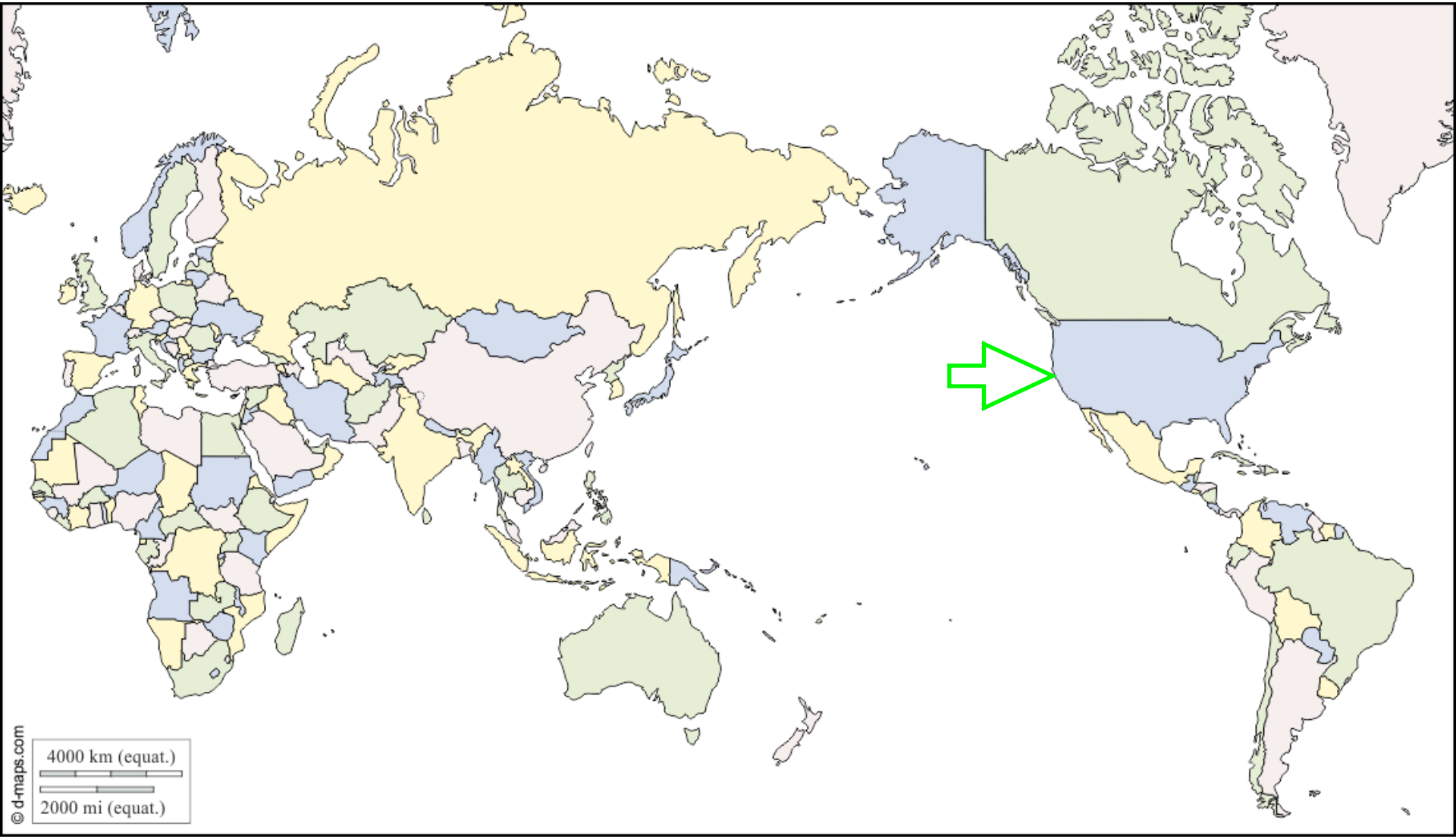
- 이진 논리식을 디지털 회로로 구현할 수 있고, 이를 바탕으로 컴퓨터에서 데이타와 명령어를 이진수로 표현하는 방법을 이해하여, 컴퓨터 프로세서가 동작하는 원리를 이해한다. 프로세서의 성능을 측정하고 비교하는 방법을 이해하고 프로세서 성능의 최적화 방안의 대표적 기법들인 파이프라인 프로세서와 메모리 계층 구조에 대해 이해한다.

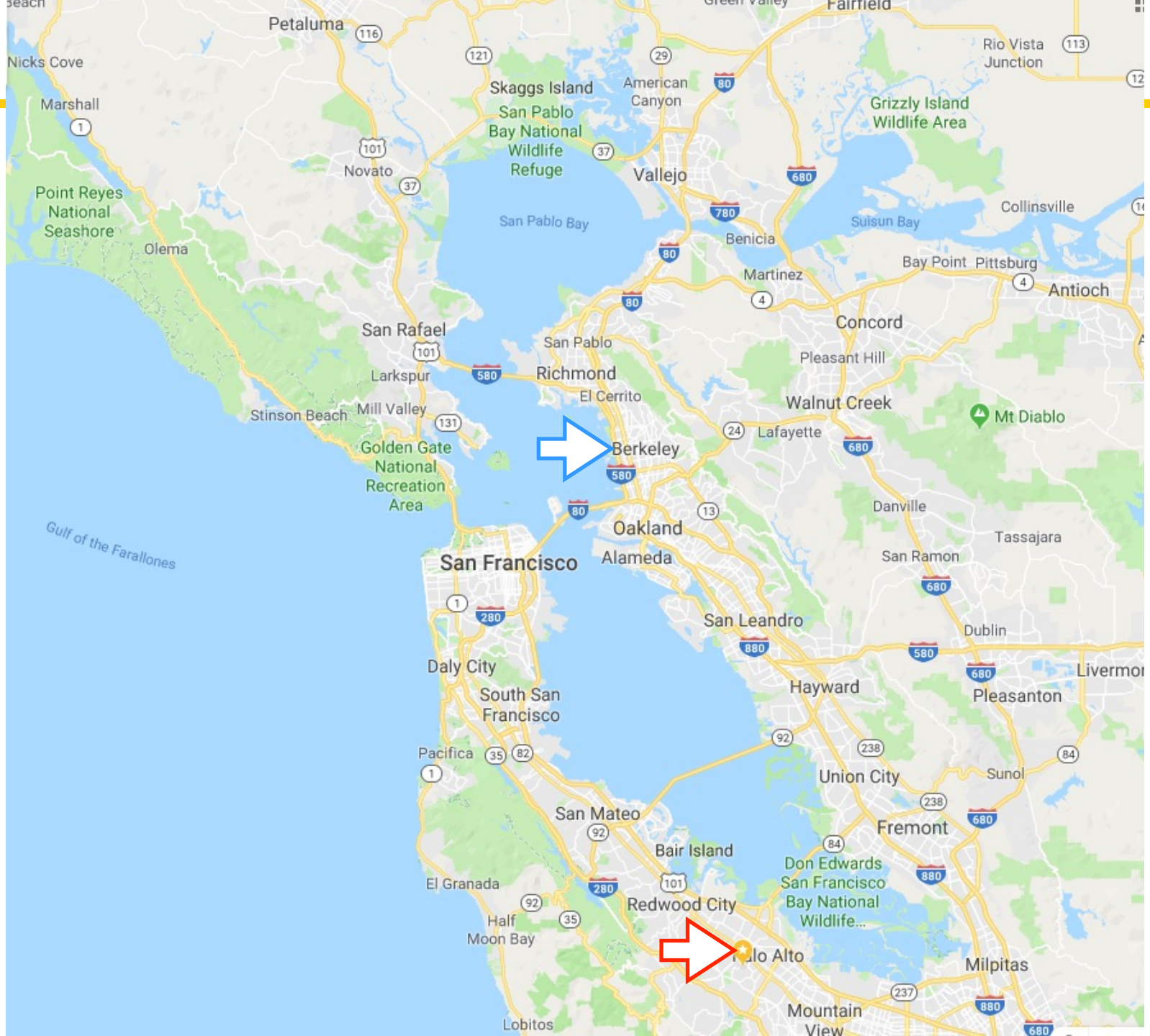
교과서

- *컴퓨터 구조 및 설계 하드웨어/소프트웨어 인터페이스, Computer Organization and Design (COD) : The Hardware/Software Interface, by* **Patterson and Hennessy**

  MIPS edition, 6th edition


SIXTH EDITION
COMPUTER ORGANIZATION AND DESIGN MIPS EDITION
The Hardware/Software Interface
컴퓨터 구조 및 설계
하드웨어/소프트웨어 인터페이스
David A. Patterson · John L. Hennessy 지음
박명순 김병기 하순회 장훈 옮김
MK
MORGAN KAUFMANN
한티미디어


4000 km (equat.)
2000 mi (equat.)
© d-maps.com


Petaluma
Nicks Cove
Marshall
Skaggs Island
San Pablo Bay National Wildlife Refuge
American Canyon
Fairfield
Rio Vista Junction
Grizzly Island Wildlife Area
Novato
Vallejo
Point Reyes National Seashore
Olema
San Pablo Bay
Benicia
Suisun Bay
Collinsville
Bay Point
Pittsburg
Antioch
Martinez
Concord
San Rafael
San Pablo
Pleasant Hill
Larkspur
Richmond
El Cerrito
Walnut Creek
Mt Diablo
Stinson Beach
Mill Valley
Lafayette
Golden Gate National Recreation Area
Berkeley
Danville
Oakland
Tassajara
Gulf of the Farallones
San Francisco
Alameda
San Ramon
San Leandro
Dublin
Daly City
Hayward
Pleasanton
South San Francisco
Pacifica
Union City
Sunol
San Mateo
Fremont
Bair Island
Don Edwards San Francisco Bay National Wildlife...
El Granada
Redwood City
Half Moon Bay
Milpitas
Mountain View
Lobitos

Author 1: David A. Patterson

- founder of RISC I & II architecture

Berkeley | EECS
Electrical Engineering and Computer Sciences

About

Home / People / Faculty / David A. Patterson

David A. Patterson

Professor Emeritus, Professor in the Graduate School

Research Areas

Computer Architecture & Engineering (ARC), Computer Architecture and Systems:

Research Centers

Center for Computational Biology (CCB)

Author 2: John L. Hennessy

- founder of MIPS architecture

JUNE 11, 2015

Stanford University President John L. Hennessy to step down in 2016

BY LISA LAPIN AND BRAD HAYWARD

President John L. Hennessy announced today that he plans to step down as Stanford University's 10th president after more than 15 years leading the transformation of one of the world's foremost research institutions.

Hennessy informed both the Board of Trustees and the Faculty Senate of his decision to depart his post in summer 2016, after serving in major academic leadership roles at Stanford for more than two decades.

"The time has come to return to

Stanford President John L. Hennessy will depart his post in summer 2016. *(Image credit: L.A. Cicero)*

Pioneers of Modern Computer Architecture Receive ACM A.M. Turing Award

Hennessy and Patterson's Foundational Contributions to Today's Microprocessors Helped Usher in Mobile and IoT Revolutions

NEW YORK, NY, March 21, 2018 – ACM, the Association for Computing Machinery, today named John L. Hennessy, former President of Stanford University, and David A. Patterson, retired Professor of the University of California, Berkeley, recipients of the 2017 ACM A.M. Turing Award for pioneering a systematic, quantitative approach to the design and evaluation of computer architectures with enduring impact on the microprocessor industry. Hennessy and Patterson created a systematic and quantitative approach to designing faster, lower power, and reduced instruction set computer (RISC) microprocessors. Their approach led to lasting and repeatable principles that generations of architects have used for many projects in academia and industry. Today, 99% of the more than 16 billion microprocessors produced annually are RISC processors, and are found in nearly all smartphones, tablets, and the billions of embedded devices that comprise the Internet of Things (IoT).

Hennessy and Patterson codified their insights in a very influential book, *Computer Architecture: A Quantitative Approach*, now in its sixth edition, reaching generations of engineers and scientists who have adopted and further developed their ideas. Their work underpins our ability to model and analyze the architectures of new processors, greatly accelerating advances in microprocessor design.

» Computer Architecture = 컴퓨터 구조

» ISA (Instruction Set Architecture) = 명령어 집합 구조

» Examples of ISA

- MIPS
- RISC I,II → SPARC
- ARM
- x86 family (intel)

Table of Contents

Card ("Green Card") 1. Pull along perforation to separate card 2. Fold bottom side (columns 3 and 4) together

MIPS Reference Data

①

CORE INSTRUCTION SET

NAME, MNEMONIC		FORMAT	OPERATION (in Verilog)		OPCODE / FUNCT (Hex)
Add	add	R	R[rd] = R[rs] + R[rt]	(1)	0 / 20_{hex}
Add Immediate	addi	I	R[rt] = R[rs] + SignExtImm	(1,2)	8_{hex}
Add Imm. Unsigned	addiu	I	R[rt] = R[rs] + SignExtImm	(2)	9_{hex}
Add Unsigned	addu	R	R[rd] = R[rs] + R[rt]		0 / 21_{hex}
And	and	R	R[rd] = R[rs] & R[rt]		0 / 24_{hex}
And Immediate	andi	I	R[rt] = R[rs] & ZeroExtImm	(3)	c_{hex}
Branch On Equal	beq	I	if(R[rs]==R[rt]) PC=PC+4+BranchAddr	(4)	4_{hex}
Branch On Not Equal	bne	I	if(R[rs]!=R[rt]) PC=PC+4+BranchAddr	(4)	5_{hex}
Jump	j	J	PC=JumpAddr	(5)	2_{hex}
Jump And Link	jal	J	R[31]=PC+8;PC=JumpAddr	(5)	3_{hex}
Jump Register	jr	R	PC=R[rs]		0 / 08_{hex}
Load Byte Unsigned	lbu	I	R[rt]={24'b0,M[R[rs] +SignExtImm](7:0)}	(2)	24_{hex}
Load Halfword Unsigned	lhu	I	R[rt]={16'b0,M[R[rs] +SignExtImm](15:0)}	(2)	25_{hex}
Load Linked	ll	I	R[rt] = M[R[rs]+SignExtImm]	(2,7)	30_{hex}
Load Upper Imm.	lui	I	R[rt] = {imm, 16'b0}		f_{hex}
Load Word	lw	I	R[rt] = M[R[rs]+SignExtImm]	(2)	23_{hex}
Nor	nor	R	R[rd] = ~ (R[rs] | R[rt])		0 / 27_{hex}
Or	or	R	R[rd] = R[rs] | R[rt]		0 / 25_{hex}
Or Immediate	ori	I	R[rt] = R[rs] | ZeroExtImm	(3)	d_{hex}
Set Less Than	slt	R	R[rd] = (R[rs] < R[rt]) ? 1 : 0		0 / $2a_{hex}$
Set Less Than Imm.	slti	I	R[rt] = (R[rs] < SignExtImm)? 1 : 0	(2)	a_{hex}
Set Less Than Imm. Unsigned	sltiu	I	R[rt] = (R[rs] < SignExtImm) ? 1 : 0	(2,6)	b_{hex}
Set Less Than Unsig.	sltu	R	R[rd] = (R[rs] < R[rt]) ? 1 : 0	(6)	0 / $2b_{hex}$
Shift Left Logical	sll	R	R[rd] = R[rt] << shamt		0 / 00_{hex}
Shift Right Logical	srl	R	R[rd] = R[rt] >>> shamt		0 / 02_{hex}
Store Byte	sb	I	M[R[rs]+SignExtImm](7:0) = R[rt](7:0)	(2)	28_{hex}
Store Conditional	sc	I	M[R[rs]+SignExtImm] = R[rt]; R[rt] = (*atomic*) ? 1 : 0	(2,7)	38_{hex}
Store Halfword	sh	I	M[R[rs]+SignExtImm](15:0) = R[rt](15:0)	(2)	29_{hex}
Store Word	sw	I	M[R[rs]+SignExtImm] = R[rt]	(2)	$2b_{hex}$
Subtract	sub	R	R[rd] = R[rs] - R[rt]	(1)	0 / 22_{hex}
Subtract Unsigned	subu	R	R[rd] = R[rs] - R[rt]		0 / 23_{hex}

(1) May cause overflow exception
(2) SignExtImm = { 16{immediate[15]}, immediate }
(3) ZeroExtImm = { 16{1b'0}, immediate }

ARITHMETIC CORE INSTRUCTION SET

②

NAME, MNEMONIC		FORMAT	OPERATION		OPCODE / FMT /FT / FUNCT (Hex)
Branch On FP True	bc1t	FI	if(FPcond)PC=PC+4+BranchAddr	(4)	11/8/1/--
Branch On FP False	bc1f	FI	if(!FPcond)PC=PC+4+BranchAddr	(4)	11/8/0/--
Divide	div	R	Lo=R[rs]/R[rt]; Hi=R[rs]%R[rt]		0/--/--/1a
Divide Unsigned	divu	R	Lo=R[rs]/R[rt]; Hi=R[rs]%R[rt]	(6)	0/--/--/1b
FP Add Single	add.s	FR	F[fd]= F[fs] + F[ft]		11/10/--/0
FP Add Double	add.d	FR	{F[fd],F[fd+1]} = {F[fs],F[fs+1]} + {F[ft],F[ft+1]}		11/11/--/0
FP Compare Single	c.x.s*	FR	FPcond = (F[fs] *op* F[ft]) ? 1 : 0		11/10/--/*y*
FP Compare Double	c.x.d*	FR	FPcond = ({F[fs],F[fs+1]} *op* {F[ft],F[ft+1]}) ? 1 : 0		11/11/--/*y*
* (*x* is eq, lt, or le) (*op* is ==, <, or <=) (*y* is 32, 3c, or 3e)					
FP Divide Single	div.s	FR	F[fd] = F[fs] / F[ft]		11/10/--/3
FP Divide Double	div.d	FR	{F[fd],F[fd+1]} = {F[fs],F[fs+1]} / {F[ft],F[ft+1]}		11/11/--/3
FP Multiply Single	mul.s	FR	F[fd] = F[fs] * F[ft]		11/10/--/2
FP Multiply Double	mul.d	FR	{F[fd],F[fd+1]} = {F[fs],F[fs+1]} * {F[ft],F[ft+1]}		11/11/--/2
FP Subtract Single	sub.s	FR	F[fd]=F[fs] - F[ft]		11/10/--/1
FP Subtract Double	sub.d	FR	{F[fd],F[fd+1]} = {F[fs],F[fs+1]} - {F[ft],F[ft+1]}		11/11/--/1
Load FP Single	lwc1	I	F[rt]=M[R[rs]+SignExtImm]	(2)	31/--/--/--
Load FP Double	ldc1	I	F[rt]=M[R[rs]+SignExtImm]; F[rt+1]=M[R[rs]+SignExtImm+4]	(2)	35/--/--/--
Move From Hi	mfhi	R	R[rd] = Hi		0 /--/--/10
Move From Lo	mflo	R	R[rd] = Lo		0 /--/--/12
Move From Control	mfc0	R	R[rd] = CR[rs]		10 /0/--/0
Multiply	mult	R	{Hi,Lo} = R[rs] * R[rt]		0/--/--/18
Multiply Unsigned	multu	R	{Hi,Lo} = R[rs] * R[rt]	(6)	0/--/--/19
Shift Right Arith.	sra	R	R[rd] = R[rt] >> shamt		0/--/--/3
Store FP Single	swc1	I	M[R[rs]+SignExtImm] = F[rt]	(2)	39/--/--/--
Store FP Double	sdc1	I	M[R[rs]+SignExtImm] = F[rt]; M[R[rs]+SignExtImm+4] = F[rt+1]	(2)	3d/--/--/--

FLOATING-POINT INSTRUCTION FORMATS

	opcode	fmt	ft	fs	fd	funct
FR	31–26	25–21	20–16	15–11	10–6	5–0
FI	31–26	25–21	20–16	15–0		

PSEUDOINSTRUCTION SET

NAME	MNEMONIC	OPERATION
Branch Less Than	blt	if(R[rs]<R[rt]) PC = Label
Branch Greater Than	bgt	if(R[rs]>R[rt]) PC = Label
Branch Less Than or Equal	ble	if(R[rs]<=R[rt]) PC = Label
Branch Greater Than or Equal	bge	if(R[rs]>=R[rt]) PC = Label
Load Immediate	li	R[rd] = immediate
Move	move	R[rd] = R[rs]

REGISTER NAME, NUMBER, USE, CALL CONVENTION

NAME	NUMBER	USE	PRESERVED ACROSS A CALL?
$zero	0	The Constant Value 0	N.A.
$at	1	Assembler Temporary	No
		Values for Function Results	