Processor Architecture

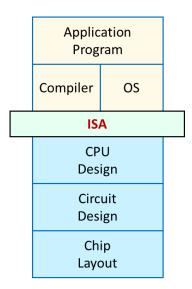
RISC-V ISA Encoding

funct7	rs2	rs1	funct3	rd	opcode	
imm[1:	:0]	rs1	funct3	rd	opcode	
imm[11:5]	rs2	rs1	funct3	imm[4:0]	opcode	
imm [10:5]	rs2	rs1	funct3	imm[4:1] imm [11]	opcode	
	imm[31:12	1		rd	opcode	
imm [20] imm[1	:1]	imm[19:1	.2]	rd	opcode	

Module Outline

- Recap: The Instruction Set Architecture
- RISC-V Instruction Formats
- RISC-V Instruction Encodings
 - R-Type Instructions
 - I-Type Instructions
 - S-Type Instructions
 - SB-Type Instructions
 - U-Type Instructions
 - UJ-Type Instructions
- Module Summary



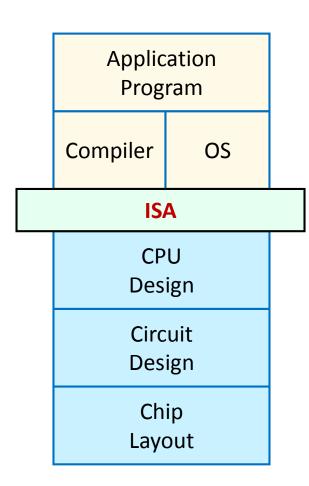


Recap

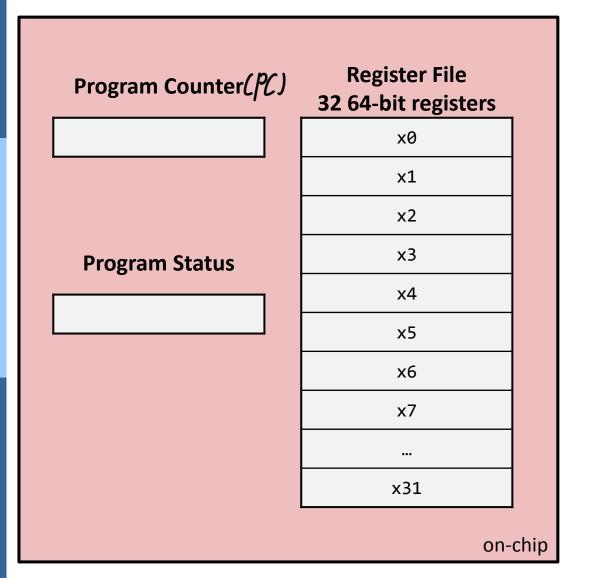
The Instruction Set Architecture

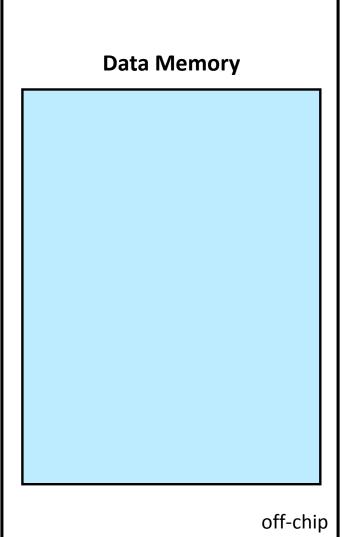
Recap - The Role of the ISA

- Assembly language view
 - Processor state
 - Registers, memory, ...
 - Instructions
 - addi, ld, jral, ...
 - How instructions are encoded as bytes
- Layer of abstraction
 - Above: how to program machine
 - Processor executes instructions in a sequence
 - Below: what needs to be built
 - Use variety of tricks to make it run fast
 - E.g., execute multiple instructions simultaneously



Recap – The (Visible) RISC-V Processor State





Recap - The (Visible) RISC-V Processor State

Program Registers

- 32 64-bit registers
- x0 hard-wired to zero
- ABI: x1: link register, x2: stack pointer

Program Counter

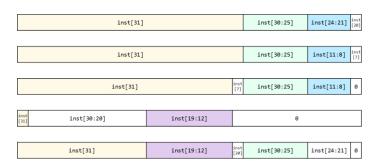
Indicates address of next instruction

Program Status

Indicates either normal operation or some error condition

Memory

- Byte-addressable storage array
- Words stored in little-endian byte order



RISC-V Instruction Formats

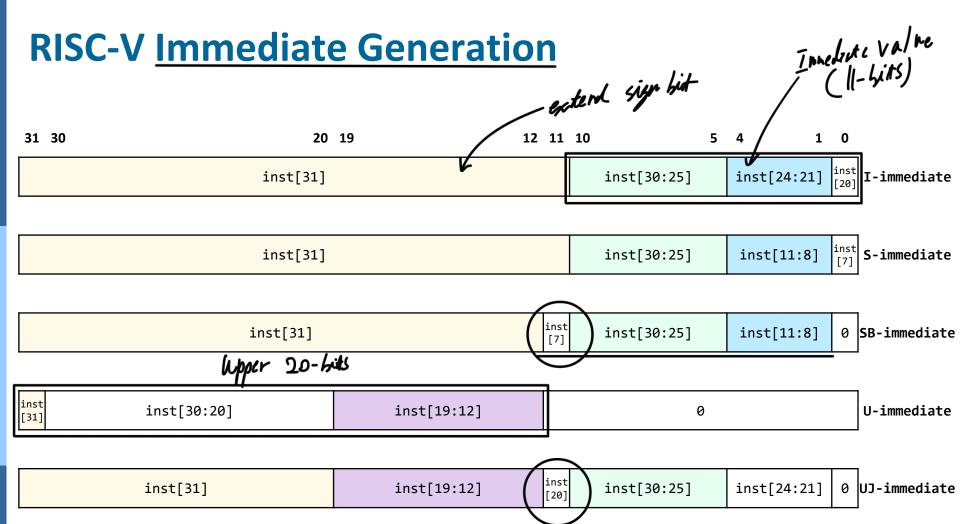
Representing Instructions

- Instructions are encoded in binary
 - Called machine code
- RISC-V instructions
 - Encoded as 32-bit instruction words
 - Small number of formats encoding operation code (opcode), register numbers, ...
 - Design principle: regularity

RISC-V Instruction Formats

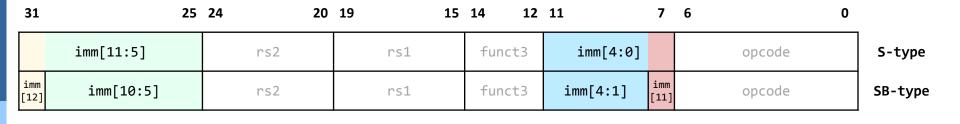
31	25	24	20	19 15	14 12	11	7	6 0	
	funct7	rs2		rs1	funct3	rd		opcode	R-type
	imm[11:0) 12-6ils		rs1	funct3	rd		opcode	I-type
	imm[11:5]	rs2		rs1	funct3	imm[4:0]		opcode	S-type
imm [12]	imm[10:5]	rs2		rs1	funct3	imm[4:1]	imm [11]	opcode	SB-type
		imm[31:12]	والزياء 20		rd		opcode	U-type
imm [20]	imm[10:1	L] [imm [11]	imm[19:1	2]	rd		opcode	UJ-type

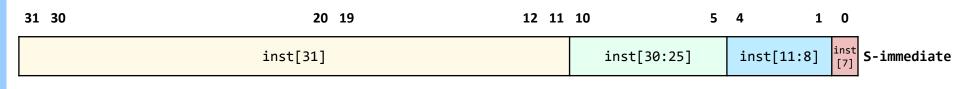
- Four basic types with two subtypes
- Type determined by opcode
- B/J-type: immediate value is encoded value shifted by 2
 - Seemingly weird immediate encoding chosen to maximize regularity



- Sign-extension is one of the most time critical operations
 - Using the <u>same bit position as the sign bit allows sign-extension</u> to proceed in parallel with instruction decoding
- B/J-type: multiplication by 2 implicit through immediate mapping
 - shift work from hardware (at-runtime) to software (compiler; before-runtime)

RISC-V Immediate Generation Example

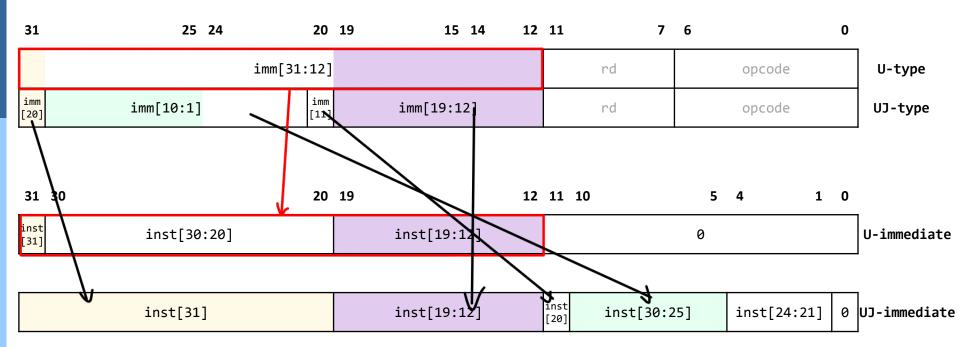




	inst[31]	inst [7]	inst[30:25]	inst[11:8]	0	SB-immediate
--	----------	-------------	-------------	------------	---	--------------

- Bit 31 of machine instruction is sign bit
- Bits 30-25 of encoding mapped to immediate at 10-5
- Bits 11-8 of encoding mapped to immediate at 4-1
- Only bit 7 is mapped to different positions (S: 0, SB: 11)
- SB-immediate: implicit multiplication by 2

RISC-V Immediate Generation Example



- Bit 31 of machine instruction is sign bit
- Bits 30-21 mapped directly (U) or moved to 10-1 (UJ)
- Bits 19-12 mapped directly
- Bit 20 is mapped directory (U) or to bit 11 (UJ)
- UJ-immediate: implicit multiplication by 2

		imm	[11	:5]					rs2					rs1			f	unc	t3		im	m[4	:0]				0	рсос	de		
		75	[11:	:5]					x7					x3			f	unc	t3		75	5[4:	0]					sd			
0	0	0	0	0	1	0	0	0	1	1	1	0	0	0	1	1	0	1	1	0	1	0	1	1	0	1	0	0	0	1	1
																_								_							
0	0	0	0	ø	1	0	0	0	1	1	1	ø	0	0	1	1	0	1	1	ø	1	0	1	1	0	1	0	ø	0	1	1
		Э				4			7	7				1				b				5			ā	1			3	3	
														0>	(047	1b6	a3			_				_				_			
														0>	a3b	671	04														

RISC-V Instruction Encodings

R-Type Instructions

31 25	24 20	19 15	14 12	11 7	6 0	
funct7	rs2	rs1	funct3	rd	opcode	R-type
additional opcode	2 nd src reg idx	1 st src reg idx	add. opcode	dst reg index	opcode	

				SEAL ANMION	K-AYPE
Instruction	Туре	Example	funct7	funct3	opcode
add	R	add rd, rs1, rs2	0000000	000	0110011
sub	R	sub rd, rs1, rs2	0100000	000	0110011
sll	R	sll rd, rs1, rs2	0000000	001	0110011
slt	R	slt rd, rs1, rs2	0000000	010	0110011
sltu	R	sltu rd, rs1, rs2	0000000	011	0110011
xor	R	xor rd, rs1, rs2	0000000	100	0110011
srl	R	srl rd, rs1, rs2	0000000	101	0110011
sra	R	sra rd, rs1, rs2	0100000 V	101	0110011
or	R	or rd, rs1, rs2	0000000	110	0110011
and	R	and rd, rs1, rs2	0000000	111	0110011

colot later

R-Type Instruction Example

sub x5, x11, x23

3	1						25	24				20	19				15	14		12	11				7	6						0	
			f	unct	t7					rs2					rs1			fı	unct	t3			rd					0	рсос	de			R-type
								-													-												
			f	unct	t7					x23					x11			fı	unct	t3			x5						sub				
6)	1	0	0	0	0	0	1	0	1	1	1	0	1	0	1	1	0	0	0	0	0	1	0	1	0	1	1	0	0	1	1	
								-																									
6)	1	0	0	0	0	0	1	0	1	1	1	0	1	0	1	1	0	0	0	0	0	1	0	1	0	1	1	0	0	1	1	bits
		4	1				1			•	7			į	5			8	3			,	2				b			:	3		nibbles
					•				•						0х	417	582	b3			•				•								encoding
_																																	

in memory

0xb3827541

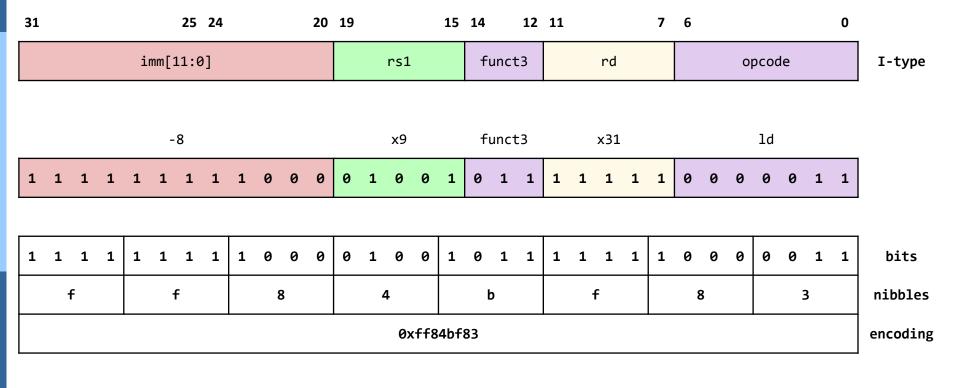
I-Type Instructions

31 25 24 20	19 15	14 12	11 7	6 0	
imm[11:0]	rs1	funct3	rd	opcode	I-type
signed 12-bit immediate/offset	src/base reg id	add. opcode	dst reg index	opcode	

Instruction	Туре	Example	inst[31:20]	funct3	opcode
addi	I	addi rd, rs1, imm12	imm12	000	0010011
slti	I	slti rd, rs1, imm12	imm12	010	0010011
sltiu	I	sltiu rd, rs1, imm12	imm12	011	0010011
xori	I	xori rd, rs1, imm12	imm12	100	0010011
ori	I	ori rd, rs1, imm12	imm12	110	0010011
andi	I	andi rd, rs1, imm12	imm12	111	0010011
slli	I	slli rd, rs1, shamt	000000 shamt	001	0010011
srli	I	srli rd, rs1, shamt	000000 shamt	101	0010011
srai	I	srai rd, rs1, shamt	010000 shamt	101	0010011
1b	I	lb rd, imm12(rs1)	imm12	000	/ 0000011
1h	I	lh rd, imm12(rs1)	imm12	001	0000011
1w	I	lw rd, imm12(rs1)	imm12	010	0000011
1d	I	ld rd, imm12(rs1)	imm12	011	0000011
1bu	I	lbu rd, imm12(rs1)	imm12	100	0000011
1hu	I	lhu rd, imm12(rs1)	imm12	101	0000011
lwu	I	lwu rd, imm12(rs1)	imm12	110	0000011
jalr	I	jalr rd, imm12(rs1)	imm12	000 new	— 1100111

I-Type Instruction Example

1d x31, -8(x9)



in memory

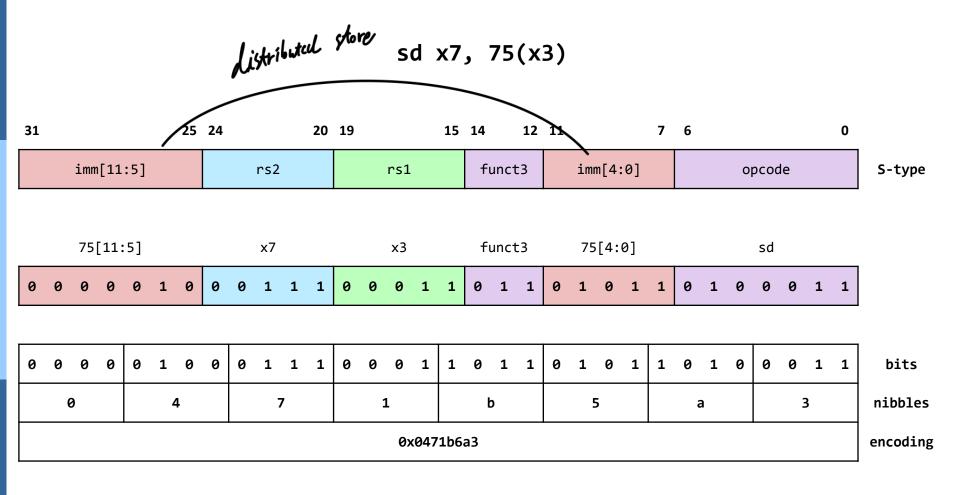
0x83bf84ff

S-Type Instructions : Save

25 24 20 19 31 15 14 12 11 7 6 0 imm[11:5] rs2 rs1 funct3 imm[4:0] opcode S-type upper 7 bits of add. lower 5 bits of 2nd src reg idx 1st src reg idx opcode signed 12-bit imm. sgd 12-bit imm. opcode

Instruction	Туре		Example	inst[31:25 11:7]	funct3	opcode
sb	S	sb	rs2, imm12(rs1)	imm12	000	0100011
sh	S	sh	rs2, imm12(rs1)	imm12	001	0100011
" SW	S	SW	rs2, imm12(rs1)	imm12	010	0100011
sd	S	sd	rs2, imm12(rs1)	imm12	011	0100011

S-Type Instruction Example

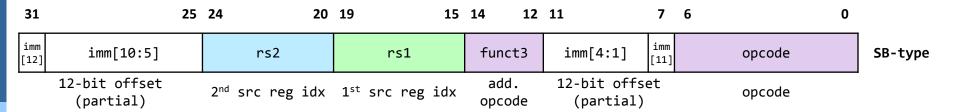


SE 컴퓨터공학부
Department of Computer Science & Engineering

in memory

0xa3b67104

SB-Type Instructions : branch



(imm[12:0] << 1) added to PC</p>

Instruction	Туре	Example	inst[31 7 30:25 4:1]	funct3	opcode
beq	SB	beq rs1, rs2, imm13	imm13 >> 1	000	1100011
bne	SB	bne rs1, rs2, imm13	imm13 >> 1	001	1100011
blt	SB	blt rs1, rs2, imm13	imm13 >> 1	100	1100011
bge	SB	bge rs1, rs2, imm13	imm13 >> 1	101	1100011
bltu	SB	bltu rs1, rs2, imm13	imm13 >> 1	110	1100011
bgeu	SB	bgeu rs1, rs2, imm13	imm13 >> 1	111	1100011

SB-Type Instruction Example

bne x10, x11, 2100 2100 = 0x834 = 0 1000 0011 010025 24 20 19 12 11 7 6 0 31 15 14 imm[10:5] funct3 imm[4:1] opcode SB-type rs2 rs1 [12] [11] 834[12], 834[10:5] x11 funct3 834[4:1], 834[11] x10 bne 0 0 1 1 0 0 0 1 0 1 bits nibbles 0 2 b 5 3 1 а е 0x02b51ae3 encoding

in memory

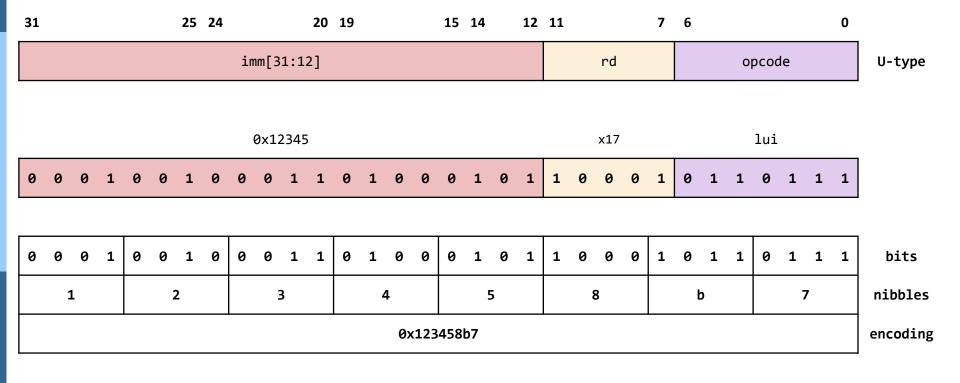
0xe31ab502

20-bils in due **U-Type Instructions** 31 25 24 20 19 15 14 12 11 7 6 0 imm[31:12] rd opcode **U-type** 20-bit constant operand (31:12) / signed offset added to PC dst reg index opcode

Instruction	Туре	Example	inst[31:12]	funct3	opcode
lui	U	lui rd, imm20	imm20	-	0110111
auipc	U	auipc rd, imm20	imm20	-	0010111

U-Type Instruction Example

lui x17, 0x12345



in memory

0xb7583412

UJ-Type Instructions

31	25 24	20	19 15 14 12	11 7	6 0	
imm [20]	imm[10:1]	imm [11]	imm[19:12]	rd	opcode	UJ-type
	signed offset	dst reg index	opcode	_		

Instruction	Туре	Example	inst[31 19:12 20 30:21]	funct3	opcode
jal	UJ	jal rd, imm21	imm21 >> 1	-	1101111

UJ-Type Instruction Example

jal x1, 0x65432

0x65432 = 0 0110 0101 0100 0011 0010

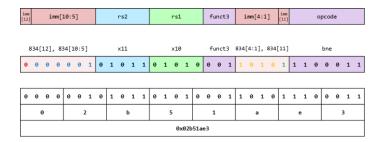
;	31						25	24				20	19				15	14		12	11				7	6						0	
i]	imm 20]				i	.mm [10:1	1]				imm [11]			in	nm[1	9:1	2]					rd					0	рсос	de			UJ-type
	0x65432 >> 1																			x1			jal										
	0	1	0	0	0	0	1	1	0	0	1	0	0	1	1	0	0	1	0	1	0	0	0	0	1	1	1	0	1	1	1	1	
																																	•
	0	1	0	0	0	0	1	1	0	0	1	0	0	1	1	0	0	1	0	1	0	0	0	0	1	1	1	0	1	1	1	1	bits
	4 3 2							(6		5 0							e f						nibbles									
	0x432650ef														encoding																		
_																																	

SE 컴퓨터공학부
Department of Computer Science & Engineering

in memory

0xef502643

Module Summary



Module Summary – RISC-V Instruction Encoding

- Designed for maximal regularity
- Four basic encoding types:
 - R, I, S, U
- Two subtypes
 - B (SB), J (UJ)
- Immediate encoding shifts work from hardware to software