



UNIVERSITI
TEKNOLOGI
PETRONAS

FINAL EXAMINATION JANUARY 2025 SEMESTER

COURSE : EDB4033 - COMPUTER SYSTEMS ARCHITECTURE
DATE : 14 APRIL 2025 (MONDAY)
TIME : 2.30 PM - 5.30 PM (3 HOURS)

INSTRUCTIONS TO CANDIDATES

1. Answer **ALL** questions in the Answer Booklet.
2. Begin **EACH** answer on a new page in the Answer Booklet.
3. Indicate clearly answers that are cancelled, if any.
4. Where applicable, show clearly steps taken in arriving at the solutions and indicate **ALL** assumptions, if any.
5. **DO NOT** open this Question Booklet until instructed.

Note :

- i. There are **FOURTEEN (14)** pages in this Question Booklet including the cover page and appendices.
- ii. **DOUBLE-SIDED** Question Booklet.

1. a. A 4-bit ripple-carry adder is to be built using the 1-bit full adder design in **FIGURE Q1**. All gates have the following propagation delays; inverters have a delay of 2 ns, 2-input AND and OR gates have a 6 ns delay, 2-input NAND and NOR gates have 8 ns delay and 3-input OR gates have a delay of 4 ns.

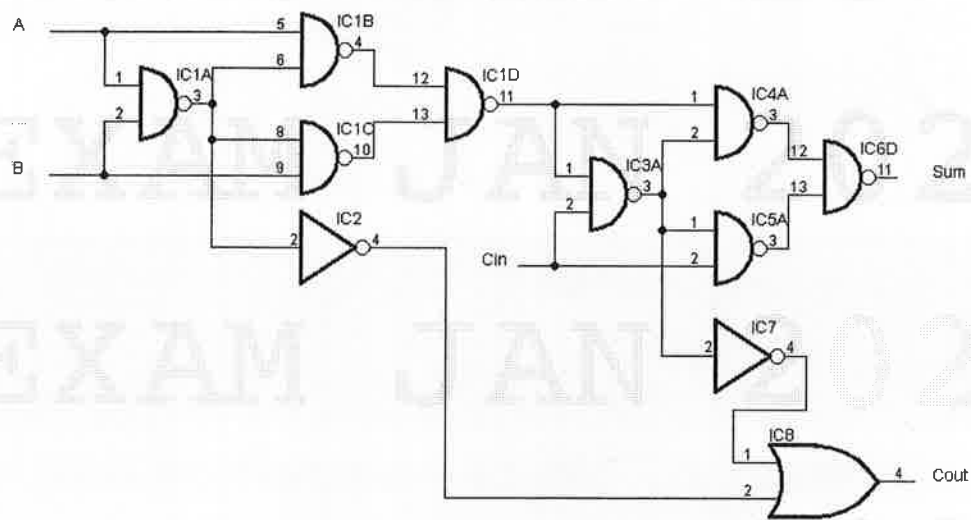


FIGURE Q1

Evaluate the worst case path or critical path for the 4-bit adder designed using the 1-bit adder design illustrated in **FIGURE Q1**.

[8 marks]

- b. Based on the RISC-V instruction set architecture and the numbered registers available in a RISC-V processor, determine if the following instructions are able to be compiled and/or able to be executed in a correct manner:

i. `beq x1 , x2, 0x1ffff`

[4 marks]

ii. `jalr 0x3ffffff`

[4 marks]

2. The single cycle RISC-V datapath illustrated in **FIGURE Q2** is capable of only running some instructions.

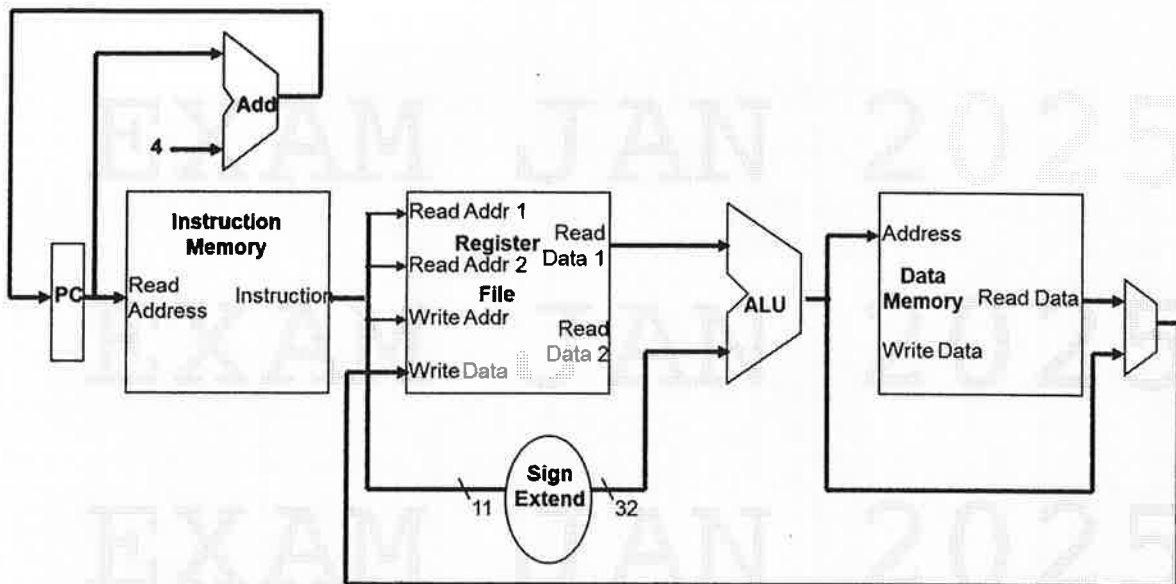


FIGURE Q2

- a. Suggest and speculate the possible type of instructions or class of instructions that can be executed on the datapath illustrated in **FIGURE Q2**.

[8 marks]

- b. A new instruction as listed in **TABLE Q2b** is to be added to the datapath illustrated in **FIGURE Q2**. Design and modify the current datapath in **FIGURE Q2** to be able to execute the following instructions in **TABLE Q2b**. Also determine the control signals needed for all instructions to be executed on this data path.

TABLE Q2b

Instruction	Instruction Operation
and rd,rs1, rs2	$\text{Reg}[\text{rd}] \leftarrow \text{Reg}[\text{rs1}] + \text{Reg}[\text{rs2}]$
lw rd, imm(rs1)	$\text{Reg}[\text{rd}] \leftarrow \text{Mem}(\text{Reg}[\text{rs1}] + \text{SignEx}[\text{Imm}])$
sw rt, Imm(rs1)	$\text{Mem}(\text{Reg}[\text{rs1}] + \text{SignEx}[\text{Imm}]) \leftarrow \text{Reg}[\text{rt}]$

[15 marks]

- c. Based on the datapath in **FIGURE Q2**, the following functional units have latencies as listed in **TABLE Q2c**.

TABLE Q2c

Functional Unit	Latency (ps)
Instruction Memory	350
Adder	100
Multiplexer	60
ALU	250
Register File	200
Data Memory	550
Program Counter	100
Sign Extender	100

List all possible critical paths in the datapath. Determine the clock cycle time required for the single cycle processor.

[5 marks]

3. In a 5-stage pipelined architecture, the following piece of code is executed:

```
lw    x5, 0(x1)
lw    x3, 8(x5)
add   x6, x5, x3
lw    x4, 8(x6)
add   x6, x6, x4
sub   x7, x4, x6
sw    x6, 8(x7)
```

- a. Analyze the above code and discuss if there are dependencies and type of data hazards.

[8 marks]

- b. Assume that the CPU is now equipped with a Hazard Detection Unit that is able to insert bubbles and stall the pipeline. Using a clock cycle table, analyze the execution of the program as it passes through the pipeline stages. Use the following standardized notation to indicate the pipeline stage for each instruction:

IF - Instruction fetch
 ID - Instruction Decode
 EX - Execution
 MEM - Memory Access
 WB - Write back

[8 marks]

- c. To further enhance the CPU, a forwarding unit was added to the Hazard Detection unit of **part (b)**. Using a clock cycle table, show the execution of the program as it passes through the pipeline stages. Evaluate the approach used for code execution using data forwarding and bubbles, and use arrows to indicate the forwarded registers and the stages where the data is being forwarded from the source stage to the destination stage. Use the same notation as in **part (b)**.

[10 marks]

4. In a computer memory system, a series of address references are applied as hexadecimal word addresses: 0x10, 0x12, 0x13, 0x14, 0x15, 0x17, 0x19, 0x1a, 0x16, 0x14, 0x13, 0x15, 0x17, 0x1c, 0x16, 0x13.
- a. Assuming that a direct mapped cache with 2 line capacity and one-word block is initially empty, label each reference in the list as a hit, compulsory miss or conflict miss and show the final contents of the cache and tags in the table in **APPENDIX I**.
- [10 marks]
- b. Assume that a four-way set-associative cache, with 2-line capacity and two-word blocks is initially empty. Label each reference in the list as a hit, compulsory miss or conflict miss and show the final contents of the cache and tags in the table in **APPENDIX II**.
- [15 marks]
- c. Determine the hit rate for each cache in **part (a)** and **part (b)**, and speculate on the cache parameters that would affect the hit rate in the cache and the limitation of these changes.

[5 marks]

-END OF PAPER-

APPENDIX I**Question 4a :**

Detach this **APPENDIX I** and attach it to your answer booklet

Examination ID : _____

Hit / Miss Identification

Address Reference	Hit / Compulsory Miss / Conflict Miss
0x10	
0x12	
0x13	
0x14	
0x15	
0x17	
0x19	
0x1a	
0x16	
0x14	
0x13	
0x15	
0x17	
0x1c	
0x16	
0x13	

APPENDIX I (continued)

Final State of Cache

Block Number	Tag	Word 0 Data
0		
1		

APPENDIX IIQuestion 4b :Detach this **APPENDIX II** and attach it to your answer booklet

Examination ID : _____

Hit / Miss Identification

Address Reference	Hit / Compulsory Miss / Conflict Miss
0x10	
0x12	
0x13	
0x14	
0x15	
0x17	
0x19	
0x1a	
0x16	
0x14	
0x13	
0x15	
0x17	
0x1c	
0x16	
0x13	

APPENDIX II (continued)**Final State of Cache**

Block Number	Set #0			Set #1		
	Tag	Word 0	Word 1	Tag	Word0	Word 1
0						
1						

Block Number	Set #2			Set #3		
	Tag	Word 0	Word 1	Tag	Word0	Word 1
0						
1						

APPENDIX III

RISC-V Green Sheet

MNEMONIC	FM	NAME	DESCRIPTION (in Verilog)	N
add, addw	R	ADD (Word)	$R[rd] \leftarrow R[rs1] + R[rs2]$	
addi, addiw	I	ADD Immediate (Word)	$R[rd] \leftarrow R[rs1] + imm$	
and	R	AND	$R[rd] \leftarrow R[rs1] \& R[rs2]$	
andi	I	AND Immediate	$R[rd] \leftarrow R[rs1] \& imm$	
auipc	U	Add Upper Immediate to PC	$R[rd] \leftarrow PC + (imm, 12'b0)$	
beq	SB	Branch Equal	$if(R[rs1] == R[rs2])$ $PC \leftarrow PC + (imm, 1b'0)$	
bge	SB	Branch Greater than or Equal	$if(R[rs1] \geq R[rs2])$ $PC \leftarrow PC + (imm, 1b'0)$	
bgeu	SB	Branch ≥ Unsigned	$if(R[rs1] \geq R[rs2])$ $PC \leftarrow PC + (imm, 1b'0)$	
blt	SB	Branch Less Than	$if(R[rs1] < R[rs2])$ $PC \leftarrow PC + (imm, 1b'0)$	
bltu	SB	Branch Less Than Unsigned	$if(R[rs1] < R[rs2])$ $PC \leftarrow PC + (imm, 1b'0)$	
bne	SB	Branch Not Equal	$if(R[rs1] \neq R[rs2])$ $PC \leftarrow PC + (imm, 1b'0)$	
csrrw	I	Cont./Stat.RegRead&Clear	$R[rd] \leftarrow CSR; CSR \leftarrow CSR \& \sim R[rs1]$	
csrrwi	I	Cont./Stat.RegRead&Clear Imm	$R[rd] \leftarrow CSR; CSR \leftarrow CSR \& \sim imm$	
csrrwui	I	Cont./Stat.RegRead&Set	$R[rd] \leftarrow CSR; CSR \leftarrow CSR R[rs1]$	
csrrwui	I	Cont./Stat.RegRead&Set Imm	$R[rd] \leftarrow CSR; CSR \leftarrow CSR imm$	
csrrw	I	Cont./Stat.RegRead&Write	$R[rd] \leftarrow CSR; CSR \leftarrow R[rs1]$	
csrrwi	I	Cont./Stat.Reg Read&Write Imm	$R[rd] \leftarrow CSR; CSR \leftarrow imm$	
ebreak	I	Environment BREAK	Transfer control to debugger	
ecall	I	Environment CALL	Transfer control to operating system	
fence	I	Synch thread	Synchronizes threads	
fence.i	I	Synch Instr & Data	Synchronizes writes to instruction stream	
jal	UJ	Jump & Link	$R[rd] \leftarrow PC+4; PC \leftarrow PC + (imm, 1b'0)$	
jalr	I	Jump & Link Register	$R[rd] \leftarrow PC+4; PC \leftarrow R[rs1] + imm$	
lb	I	Load Byte	$R[rd] \leftarrow \{56'bM[(7), M[R[rs1] + imm](7:0)]\}$	
lbu	I	Load Byte Unsigned	$R[rd] \leftarrow \{56'b0, M[R[rs1] + imm](7:0)\}$	
ld	I	Load Doubleword	$R[rd] \leftarrow M[R[rs1] + imm](63:0)$	
lh	I	Load Halfword	$R[rd] \leftarrow \{48'bM[(15), M[R[rs1] + imm](15:0)]\}$	
lhu	I	Load Halfword Unsigned	$R[rd] \leftarrow \{48'b0, M[R[rs1] + imm](15:0)\}$	
lui	UJ	Load Upper Immediate	$R[rd] \leftarrow \{32'bimm < 31>, imm, 12'b0\}$	
lw	I	Load Word	$R[rd] \leftarrow \{32'bM[(31), M[R[rs1] + imm](31:0)]\}$	
lwu	I	Load Word Unsigned	$R[rd] \leftarrow \{32'b0, M[R[rs1] + imm](31:0)\}$	
or	R	OR	$R[rd] \leftarrow R[rs1] R[rs2]$	
ori	I	OR Immediate	$R[rd] \leftarrow R[rs1] imm$	
sb	S	Store Byte	$M[R[rs1] + imm](7:0) \leftarrow R[rs2](7:0)$	
sd	S	Store Doubleword	$M[R[rs1] + imm](63:0) \leftarrow R[rs2](63:0)$	
sh	S	Store Halfword	$M[R[rs1] + imm](15:0) \leftarrow R[rs2](15:0)$	
sl, sliw	R	Shift Left (Word)	$R[rd] \leftarrow R[rs1] \ll R[rs2]$	
slai, slilw	I	Shift Left Immediate (Word)	$R[rd] \leftarrow R[rs1] \ll imm$	
slt	R	Set Less Than	$R[rd] \leftarrow (R[rs1] < R[rs2]) ? 1 : 0$	
slti	I	Set Less Than Immediate	$R[rd] \leftarrow (R[rs1] < imm) ? 1 : 0$	
sltiu	I	Set < Immediate Unsigned	$R[rd] \leftarrow (R[rs1] < imm) ? 1 : 0$	
sltui	R	Set Less Than Unsigned	$R[rd] \leftarrow (R[rs1] < R[rs2]) ? 1 : 0$	
sra, sraw	R	Shift Right Arithmetic (Word)	$R[rd] \leftarrow R[rs1] \ggg R[rs2]$	
srai, srawi	I	Shift Right Arith Imm (Word)	$R[rd] \leftarrow R[rs1] \ggg imm$	
srl, srlw	R	Shift Right (Word)	$R[rd] \leftarrow R[rs1] \gg R[rs2]$	
sri, srlw	I	Shift Right Immediate (Word)	$R[rd] \leftarrow R[rs1] \gg imm$	
sub	R	SUBtract (Word)	$R[rd] \leftarrow R[rs1] - R[rs2]$	
sw	S	Store Word	$M[R[rs1] + imm](31:0) \leftarrow R[rs2](31:0)$	
xor	R	XOR	$R[rd] \leftarrow R[rs1] \oplus R[rs2]$	
xori	I	XOR Immediate	$R[rd] \leftarrow R[rs1] \oplus imm$	

OPCODES IN NUMERICAL ORDER BY OPCODE

MNEMONIC	FM	OPCODE	FUNCT3	FUNCT7 OR IMM	HEXADECIM
lb	I	0000011	000		03/0
lh	I	0000011	001		03/1
lw	I	0000011	010		03/2
ld	I	0000011	011		03/3
lbu	I	0000011	100		03/4
lhu	I	0000011	101		03/5
lwa	I	0000011	110		03/6
fence	I	0001111	000		0F/0
fence.i	I	0001111	001		0F/1
addi	I	0010011	000		13/0
slli	I	0010011	001	0000000	13/1/00
slli	I	0010011	010		13/2
slliu	I	0010011	011		13/3
xori	I	0010011	100		13/4
xori	I	0010011	101	0000000	13/5/00
srai	I	0010011	101	0100000	13/5/20
ori	I	0010011	110		13/6
andi	I	0010011	111		13/7
addipc	U	0010111			17
addiw	I	0011011	000		1B/0
slliw	I	0011011	001	0000000	1B/1/00
slliw	I	0011011	101	0000000	1B/5/00
sraiw	I	0011011	101	0100000	1B/5/20
sb	S	0100011	000		23/0
sh	S	0100011	001		23/1
sw	S	0100011	010		23/2
sd	S	0100011	011		23/3
sdd	R	0110011	000	0000000	33/0/00
sdb	R	0110011	000	0100000	33/0/20
sll	R	0110011	001	0000000	33/1/00
sll	R	0110011	010	0000000	33/2/00
sllr	R	0110011	011	0000000	33/3/00
xor	R	0110011	100	0000000	33/4/00
srl	R	0110011	101	0000000	33/5/00
sra	R	0110011	101	0100000	33/5/20
or	R	0110011	110	0000000	33/6/00
and	R	0110011	111	0000000	33/7/00
lui	I	0110011			37
addw	R	0110011	000	0100000	3B/0/00
sllw	R	0110011	001	0100000	3B/0/20
sllw	R	0110011	010	0000000	3B/1/00
sllw	R	0110011	011	0000000	3B/1/20
sra	R	0110011	101	0100000	3B/5/20
lwr	SB	1100011			63/0
lwr	SB	1100011	1		63/1
lwr	SB	1100011	2		63/2
lwr	SB	1100011	3		63/3
lwr	SB	1100011	4		63/4
lwr	SB	1100011	5		63/5
lwr	SB	1100011	6		63/6
lwr	SB	1100011	7		63/7
lwr	SB	1100011	8		63/8
lwr	SB	1100011	9		63/9
lwr	SB	1100011	10		63/10
lwr	SB	1100011	11		63/11
lwr	SB	1100011	12		63/12
lwr	SB	1100011	13		63/13
lwr	SB	1100011	14		63/14
lwr	SB	1100011	15		63/15
lwr	SB	1100011	16		63/16
lwr	SB	1100011	17		63/17
lwr	SB	1100011	18		63/18
lwr	SB	1100011	19		63/19
lwr	SB	1100011	20		63/20
lwr	SB	1100011	21		63/21
lwr	SB	1100011	22		63/22
lwr	SB	1100011	23		63/23
lwr	SB	1100011	24		63/24
lwr	SB	1100011	25		63/25
lwr	SB	1100011	26		63/26
lwr	SB	1100011	27		63/27
lwr	SB	1100011	28		63/28
lwr	SB	1100011	29		63/29
lwr	SB	1100011	30		63/30
lwr	SB	1100011	31		63/31
lwr	SB	1100011	32		63/32
lwr	SB	1100011	33		63/33
lwr	SB	1100011	34		63/34
lwr	SB	1100011	35		63/35
lwr	SB	1100011	36		63/36
lwr	SB	1100011	37		63/37
lwr	SB	1100011	38		63/38
lwr	SB	1100011	39		63/39
lwr	SB	1100011	40		63/40
lwr	SB	1100011	41		63/41
lwr	SB	1100011	42		63/42
lwr	SB	1100011	43		63/43
lwr	SB	1100011	44		63/44
lwr	SB	1100011	45		63/45
lwr	SB	1100011	46		63/46
lwr	SB	1100011	47		63/47
lwr	SB	1100011	48		63/48
lwr	SB	1100011	49		63/49
lwr	SB	1100011	50		63/50
lwr	SB	1100011	51		63/51
lwr	SB	1100011	52		63/52
lwr	SB	1100011	53		63/53
lwr	SB	1100011	54		63/54
lwr	SB	1100011	55		63/55
lwr	SB	1100011	56		63/56
lwr	SB	1100011	57		63/57
lwr	SB	1100011	58		63/58
lwr	SB	1100011	59		63/59
lwr	SB	1100011	60		63/60
lwr	SB	1100011	61		63/61
lwr	SB	1100011	62		63/62
lwr	SB	1100011	63		63/63
lwr	SB	1100011	64		63/64
lwr	SB	1100011	65		63/65
lwr	SB	1100011	66		63/66
lwr	SB	1100011	67		63/67
lwr	SB	1100011	68		63/68
lwr	SB	1100011	69		63/69
lwr	SB	1100011	70		63/70
lwr	SB	1100011	71		63/71
lwr	SB	1100011	72		63/72
lwr	SB	1100011	73		63/73
lwr	SB	1100011	74		63/74
lwr	SB	1100011	75		63/75
lwr	SB	1100011	76		63/76
lwr	SB	1100011	77		63/77
lwr	SB	1100011	78		63/78
lwr	SB	1100011	79		63/79
lwr	SB	1100011	80		63/80
lwr	SB	1100011	81		63/81
lwr	SB	1100011	82		63/82
lwr	SB	1100011	83		63/83
lwr	SB	1100011	84		63/84
lwr	SB	1100011	85		63/85
lwr	SB	1100011	86		63/86
lwr	SB	1100011	87		63/87
lwr	SB	1100011	88		63/88
lwr	SB	1100011	89		63/89
lwr	SB	1100011	90		63/90
lwr	SB	1100011	91		63/91
lwr	SB	1100011	92		63/92
lwr	SB	1100011	93		63/93
lwr	SB	1100011	94		63/94
lwr	SB	1100011	95		63/95
lwr	SB	1100011	96		63/96
lwr	SB	1100011	97		63/97
lwr	SB	1100011	98		63/98
lwr	SB	1100011	99		63/99
lwr	SB	1100011	100		63/100
lwr	SB	1100011	101		63/101
lwr	SB	1100011	102		63/102
lwr	SB	1100011	103		63/103
lwr	SB	1100011	104		63/104
lwr	SB	1100011	105		63/105
lwr	SB	1100011	106		63/106
lwr	SB	1100011	107		63/107
lwr	SB	1100011	108		63/108
lwr	SB	1100011	109		63/109
lwr	SB	1100011	110		63/110
lwr	SB	1100011	111		63/111
lwr	SB	1100011	112		63/112
lwr	SB	1100011	113		63/113
lwr	SB	1100011	114		63/114
lwr	SB	1100011	115		63/115
lwr	SB	1100011	116		63/116
lwr	SB	1100011	117		63/117
lwr	SB	1100011	118		63/118
lwr	SB	1100011	119		63/119
lwr	SB	1100011	120		63/120
lwr	SB	1100011	121		63/121
lwr	SB	1100011	122		63/122
lwr	SB	1100011	123		63/123
lwr	SB	1100011	124		63/124
lwr	SB	1100011	125		63/125
lwr	SB	1100011	126		63/126
lwr	SB	1100011	127		63/127
lwr	SB	1100011	128		63/128
lwr	SB	1100011	129		63/129
lwr	SB	1100011	130		63/130
lwr	SB	1100011	131		63/131
lwr	SB	1100011	132		63/132
lwr	SB	1100011	133		63/133
lwr	SB	1100011	134		63/134
lwr	SB	1100011	135		63/135
lwr	SB	1100011	136		63/136
lwr	SB	1100011	137		63/137
lwr	SB	1100011	138		63/138
lwr	SB	1100011	139		63/139
lwr	SB	1100011	140		63/140
lwr	SB	1100011	141		63/141
lwr	SB	1100011	142		63/142
lwr	SB	1100011	143		63/143
lwr	SB	1100011	144		63/144
lwr	SB	1100011	145		63/145
lwr	SB	1100011	146		63/146
lwr	SB	1100011	147		63/147
lwr	SB	1100011	148		63/148
lwr	SB	1100011	149		63/149
lwr	SB	1100011	150		63/150
lwr	SB	1100011	151		63/151
lwr	SB	1100011	152		63/152
lwr	SB	1100011	153		63/153
lwr	SB	1100011	154		63/154
lwr	SB	1100011	155		63/155
lwr	SB	1100011	156		63/156
lwr	SB	1100011	157		63/157
lwr	SB	1100011	158		63/158
lwr	SB	1100011	159		63/159
lwr	SB	1100011	160		63/160
lwr	SB	1100011	161		63/161
lwr	SB	1100011	162		63/162
lwr	SB	1100011	163		63/163
lwr	SB	1100011	164		63/164
lwr	SB	1100011	165		63/165
lwr	SB	1100011	166		63/166
lwr	SB	1100011	167		63/167
lwr	SB	1100011	168		63/168
lwr	SB	1100011	169		63/169
lwr	SB	1100011	170		63/170
lwr	SB	1100011	171		63/171
lwr	SB	1100011	172		63/172
lwr	SB	1100011	173		63/173
lwr	SB	1100011	174		63/174
lwr	SB	1100011	175		63/175
lwr	SB	1100011	176		63/176
lwr	SB	1100011	177		63/177
lwr	SB	1100011	178		63/178
lwr	SB	1100011	179		63/179
lwr	SB	1100011	180		63/180
lwr	SB	1100011	181		63/181
lwr	SB	1100011	182		63/182
lwr	SB	1100011	183		63/183
lwr	SB	1100011	184		63/184
lwr	SB	1100011	185		63/185
lwr	SB	1100011	186		63/186
lwr	SB	1100011	187		63/187
lwr	SB	1100011	188		63/188
lwr	SB	1100011	189		63/189
lwr	SB	1100011	1		

CORE INSTRUCTION FORMATS

	31	27	26	25	24	20	19	15	14	12	11	7	6	0
R	funct7				rs2		rs1		funct3		rd		Opcode	
I	imm[11:0]						rs1		funct3		rd		Opcode	
S	imm[11:5]				rs2		rs1		funct3		imm[4:0]		opcode	
SB	imm[12 10:5]				rs2		rs1		funct3		imm[4:1 11]		opcode	
U	imm[31:12]										rd		opcode	
UJ	imm[20 10:1 11 19:12]										rd		opcode	

REGISTER NAME, USE, CALLING CONVENTION**④**

REGISTER	NAME	USE	SAVER
x0	zero	The constant value 0	N.A.
x1	ra	Return address	Caller
x2	sp	Stack pointer	Callee
x3	gp	Global pointer	--
x4	tp	Thread pointer	--
x5-x7	t0-t2	Temporaries	Caller
x8	s0/fp	Saved register/Frame pointer	Callee
x9	s1	Saved register	Callee
x10-x11	a0-a1	Function arguments/Return values	Caller
x12-x17	a2-a7	Function arguments	Caller
x18-x27	s2-s11	Saved registers	Callee
x28-x31	t3-t6	Temporaries	Caller
f0-f7	ft0-ft7	FP Temporaries	Caller
f8-f9	fs0-fs1	FP Saved registers	Callee
f10-f11	fa0-fa1	FP Function arguments/Return values	Caller
f12-f17	fa2-fa7	FP Function arguments	Caller
f18-f27	fs2-fs11	FP Saved registers	Callee
f28-f31	ft8-ft11	$R[rd] = R[rs1] + R[rs2]$	Caller