6.2 Memory Organisation

- Each address contains 1 byte = 8 bit of content.
- Memory addresses are 32-bit long (2^{30} memory words).
- 32 registers, each 4-byte long. Each word is also 4-byte long. (Note that words are usually 2^n bytes)

6.3 MIPS Instruction Classification

6.3.1 R-format

- `op $rd, $rs, $rt`
- `sll $rd, $rt, shamt` (rs = 0)

6.3.2 I-format

- `op $rt, $rs, Immediate`
- Immediate is a 16 bit 2s complement constant
- Displacement address: offset from address in rs
- PC-relative address: no of instructions from next instruction $PC = PC + 4 + (Immediate \times 4)$

6.3.3 J-format (can jump up to 256MB range)

- `op Immediate`
- pseudo-direct address: remove last 2 bit (since word-aligned, by default the 2 least significant bits are 00) and 4 most significant bits (always the same as instruction address).
- eg xxxx00001111000011110000111100~~00~~, immediate is 00001111000011110000111100

7 Instruction Set Architecture

For modern processors: **General-Purpose Register** (GPR) is most common. **RISC** typically uses **Register-Register (Load/Store)** design, e.g. MIPS, ARM. **CISC** use a mixture of Register-Register and Register-Memory, e.g. IA32

7.1 Data Storage

- **Stack architecture** : Operands are implicitly on top of the stack.
- **Accumulator architecture** : One operand is implicitly in the accumulator (a special register)
- **General-purpose register architecture** : only explicit operands
- **Register-memory architecture** : one operand in memory.
- **Register-register (or load store) architecture**
- **Memory-memory architecture** : all operands in memory.

7.2 Memory Addressing Modes

- **Endianness** : Relative ordering of bytes in a multiple-byte word stored in memory
- **Big-endian** : **Most** significant byte stored in lowest address
- **Little-endian** : **Least** significant byte stored in lowest address ("reverse-order")
- **Addressing modes** : in MIPS, only 3: **Register** add $t1, $t2, $t3, **Immediate** addi $t1, $t2, 98, **Displacement** lw $t1, 20($t2)

7.3 Operations in the instruction set

Amdahl's law: make common cases fast. Optimise frequently used instructions (**Load**: 22%, **Conditional Branch**: 20%, **Compare** 16%, **Store**: 12%)

7.4 Instruction Formats

- **Instruction Length** :
- **Variable-length instructions** : Require multi-step fetch and decode. Allow for a more flexible (but complex) and compact instruction set.
- **Fixed-length instructions** : used in most RISC, e.g. MIPS instructions are 4-bytes long. Allow for easy fetch and decode, simplify pipelining and parallelism. Instruction bits are scarce.
- **Hybrid instructions** : a mix of variable- and fixed-length instructions.
- **Instruction Fields** : **opcode** (unique code to specify the desired operation) and **operands** (zero or more additional information needed for the operation)

7.5 Encoding the Instruction Set

- **Expanding Opcode** scheme:
- E.g. **Type-A**: **6-bit** opcode, **Type-B**: **11-bits** opcode. Max no of instructions = $1 + (2^6 - 1) \times 2^5 = 2017$

(1 Type-A instruction, Type-B "steals" $[2^6 - 1]$ opcodes from Type-A to prefix, each prefix having $[2^{11-6} = 2^5]$ opcodes)

8 Datapath

8.1 Instruction Execution Cycle

For MIPS: (1)Fetch (2)Decode & Operand Fetch (3)ALU (4)Memory Access (5)Result Write

- **Fetch** : Get instruction from memory, address is in Program Counter (PC) Register
- **Decode** : Find out the operation required
- **Operand Fetch** : Get operand(s) needed for operation
- **Execute** : Perform the required operation
- **Result Write (Store)** : Store the result of the operation

8.2 Elements

- **Adder** **Input**: two 32-bit numbers, **Output**: sum of input numbers
- **Register File** **Input**: three 5-bit: Read register 1, Read register 2, Write register; 32-bit Write data, **Output**: two 32-bit Read data 1, Read data 2; **Control**: 1-bit RegWrite (1 = write)
- **Multiplexer** **Input**: n lines of same width, **Control**: m bits where $n = 2^m$, **Output**: Select i^{th} input line if control = i
- **Arithmetic Logic Unit** : **Input**: two 32-bit numbers, **Control**: 4-bit to decide the particular operation, **Output**: 32-bit ALU result, 1-bit isZero?

ASCII Table

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	|
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

Positive Power of 2

Exp	Val	Exp	Val	Exp	Val	Exp	Val
2^{0}	1	2^{8}	256	2^{16}	65,536	2^{24}	16,777,216
2^{1}	2	2^{9}	512	2^{17}	131,072	2^{25}	33,554,432
2^{2}	4	2^{10}	1,024	2^{18}	262,144	2^{26}	67,108,864
2^{3}	8	2^{11}	2,048	2^{19}	524,288	2^{27}	134,217,728
2^{4}	16	2^{12}	4,096	2^{20}	1,048,576	2^{28}	268,435,456
2^{5}	32	2^{13}	8,192	2^{21}	2,097,152	2^{29}	536,870,912
2^{6}	64	2^{14}	16,384	2^{22}	4,194,304	2^{30}	1,073,741,824
2^{7}	128	2^{15}	32,768	2^{23}	8,388,608	2^{31}	2,147,483,648

Negative Power of 2

Exp	Val	Exp	Val
2^{-1}	0.5	2^{-9}	0.001953125
2^{-2}	0.25	2^{-10}	0.0009765625
2^{-3}	0.125	2^{-11}	0.00048828125
2^{-4}	0.0625	2^{-12}	0.000244140625
2^{-5}	0.03125	2^{-13}	0.0001220703125
2^{-6}	0.015625	2^{-14}	0.00006103515625
2^{-7}	0.0078125	2^{-15}	0.000030517578125
2^{-8}	0.00390625	2^{-16}	0.0000152587890625