Notation

- n_x: number of x
- P_x: percentage or power of x
- T_x: time of x
- i: instructions

Basic conecpt

Speed Up Ratio: $\frac{T_{Prev}}{T_{Current}} = \frac{\text{Throughput current}}{\text{Throughput previous}}$
SPECratio $= \frac{T_{ref}}{T_{current}}$, where reference times means baseline time in an **benchmark**
The execution time imporved $= \frac{T_{Prev} - T_{Current}}{T_{Prev}}$
Throughput: $\frac{n_{data}}{T_{exec}}$, maxIPS: $\frac{n_{instruction}}{T_{exec}}$
program maxIPS $= \sum_{i \in prog} maxIPS_i \times P_i$
Performance Per Wat $= \frac{maxIPS}{Power}$, Not Throughtput
MIPS: Millions of Instructions Per Second

Clock Rate/Time/CPI

CPI: Average cycles per instruction, **not reverse**

$$\text{CPI} = \frac{n_{cycles}}{n_{instruct}} \leftrightarrow n_{cycles} = CPI \times n_{instruct}$$

$$T_{execution} = CPI \times n_{instruct} \times T_{clock} = \frac{CPI \times n_{instruct}}{\text{CLK Rate}}$$

Clock Rate vs Clock Time are Reciprocal

$$\text{CLK Rate} = \frac{n_{cycle}}{T_{execution}}, \text{CLK Time} = \frac{T_{execution}}{n_{cycle}}$$

Clock Rate vs Clock Time's Unit

- 1 kHz $= 10^3$ Hz
- 1 MHz $= 10^6$ Hz
- 1 GHz $= 10^9$ Hz
- 1 s $= 10^3$ ms
- 1 s $= 10^6\ \mu s$
- 1 s $= 10^9\ ns$

Problem Template for Clock Cycle

1: Run a benchmark on different HW, different in CPI/$n_{instruct}$/CLK Rate
2: Bearkdown a program by to FP,INT,L/S,Branch, change attributes accordingly **Notice**:

- Use table to record data of HW or instructions
- Use the right formula $\frac{CPI \times n_{instruct}}{\text{CLK Rate}}$
- Different in Speed Up Ratio and Time saved
- Some speed up ratio may not achieved IRL

Eg: 1-1, 1-6, 1-10

Amdahl's law

$S = \frac{1}{(1-P) + \frac{P}{N}}$, P is Parallel ratio and N is P speed up
In some probles, we have Parallel overhead
Eg: 1-2, 1-5, 1-11

Power and Energy of HW

Power $= C \times V^2 \times f$ W
Energy $= C \times V^2$ J

Power and Energy of Problem Template

Shut down some machine/Change chip attribute/Change running state(aka. power comsumption) and Calculate power/energy change, note:

- Shut down P % machine → save P % energy
- Note power/energy change are different, where **energy has no relation with frenquent**

Eg: 1-7

QoS

MTTF: Mean Time To Failure, $MTTF = \frac{1}{FIT}$
FIT:Failure In Time, $FIT_{system} = \frac{FIT_{single}}{P_{systemfail}}$ TODO
Eg: 1-4

Moore's Law and the Power Wall

Moore's Law: number of transistors on a microchip doubles approximately every two years, exponential.
Power Wall refers to a limitation in computer architecture related to power consumption and heat dissipation, causing Moore's Law no longer work.
Multicore architects doing with the extra transistors now to increase performance Eg:1-8

RISCV Translation Basic

a/b/c:value in register, A/B/C:address in register

- `c=a-b` : sub x,a,b
- `c=a+1` : addi x,a,1
- `A=a` : sw a, 0(A)
- `b=B` : lw b, 0(B)
- `a=a<<2` or `a=a*4`: slli a,a,2

Eg: 2-1

RISCV Translation Advance

Indexing: `slli` → add start and offset → lw/sw

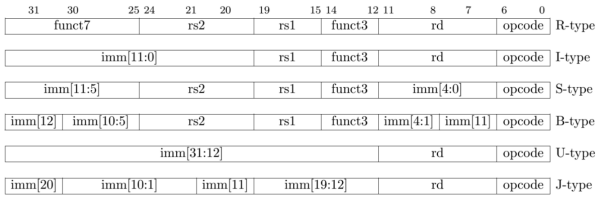
$$x_1 = A[\underbrace{\underbrace{i}_{\texttt{slli i, i,2}}}_{\texttt{add i, A, i}}]_{\texttt{lw x1, 0(i)}} \qquad \underbrace{A[\underbrace{i}_{\texttt{slli i, i,2}}]}_{\texttt{add i, A, i}} = x_1{}_{\texttt{sw x1, 0(i)}}$$

Loop: couter init → LOOP tag/branch → counter step → loop body → jump back/end tag

`int i = 10;`	`addi x1, 10, x0`
`while (i >= 0)`	`LOOP:beq x1, x0, DONE`
`i--`	`addi x1, x1, -1`
(Loop Body)	(Loop Body)
} (end of loop)	`jal x0, LOOP`
(end of loop)	`DONE`

Eg: 2-1, 2-5, 2-6

RISCV format



Eg: 2-2

RISCV format Problem Template

Given instruction, check it's binary code, Eg: 2-3
Since imm field is limit, calculate some upper/lower bound, Eg: 2-5
Imm

- I: 12bits, `addi`'s imm value's bound, unsign/signed
- S: 12 bits, `sw`'s offset's bound, raw address
- B: 13 bits, raw address
- U: large imm, 20bits
- J: range to jump, 21bits

B and J imm layout: both of them has no 0 bits, that's because the target address is always 2-byte aligned(32bits), the last byte is meaningless.

Jump and Branch

Shift 4n in binary(default) = shift n in hex

Shift meaning: $\underset{\text{shift}}{s}$ $\overset{\text{left or right}}{r}$ $\underset{\text{logic or arithmetic}}{l}$ $\overset{\text{immediate}}{i}$

Logical: and/andi, or/ori, xor/xori ; Note: andi = select bits, ori ≈ add 2 binary , xori 0xFF = not

Eg: 2-2, 2-4

Bit op

Jump: jal(address=imm), jalr(address=reg+imm)

Branch: `b<condition>`, if condition==true jump

subtype: beq, bne, blt, bge(signed), xxx u(unsigned)

ORDER `blt, rs1, rs2, Label`: if $rs1 < rs2$ jump

Eg: 2-6

Loop Instruction Counts

$$n_i = n_{loop} \times n_{\text{i in loop}} + n_{\text{i out loop}} + \underbrace{1}_{\text{jump out of loop}}$$

Eg: 2-6, 2-7

Big/Little Endian

Memory	0	1	2	3	0x12345678
big endian	12	34	56	78	MSB in low address
little endian	78	56	34	12	LSB in low address

Note: both memory and data are in hex, Eg: 2-8

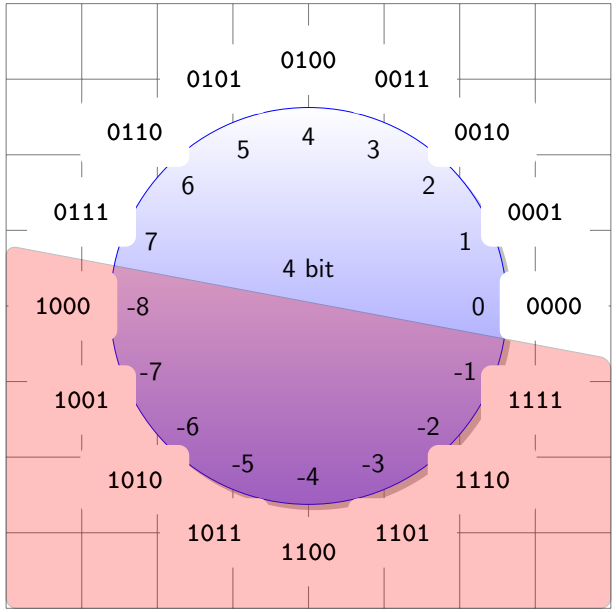
U instruction

Load Upper Immediate `lui` or Add Upper Immediate to PC `auipc`, Eg: 2-9(load 64 bits)

To load 0xABCD1234(32bits) to x10:

```
lui x10, 0xABCD1 // [31:12]
addi x10, x10, 0x234 // [11:0]
```

Two's complement and valid range



Since register are 64 bits, the range of int/uint is int: $[2^{-31}, 2^{31} - 1]$, uint: $[0, 2^{32} - 1]$, Eg: 2-10