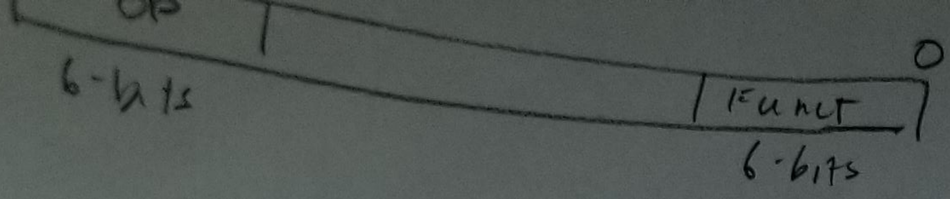



$2^6 = 64$ instructions can be held in OP and Funct each

So $64 \cdot 2 =$ 128 total instructions

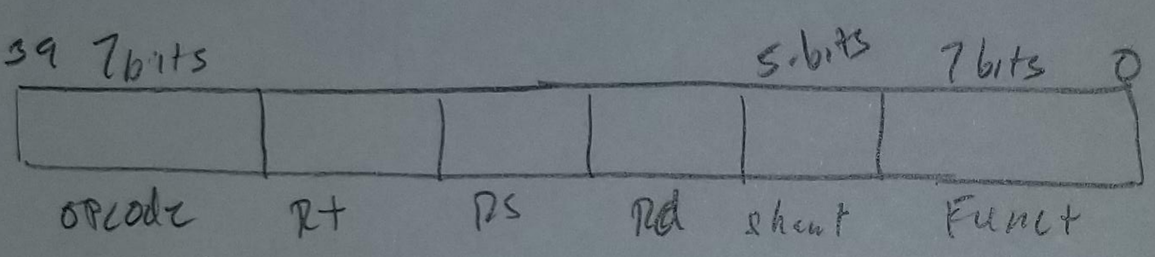
---

①

Increments the index by the size of an int (4 bytes), and then stores that value, at A[index+1], into R$\phi$. It then adds the contents of R1$\phi$ and R5$\phi$, which are A[index] + A[index+1] and stores the result in R2$\phi$. R2$\phi$ is then stored in register R12. So R2$\phi$ = A[index] + A[index+1].

---

②




SHAMT still 5 bits

So... 40-bit word size
- 14-bit opcode/funct
- 5-bit SHAMT

21 bits remaining

200 instructions since splitting opcode to hold 0-99 in opcode or Funct

$2^6 < 100 < 2^7$ - 7-bits for opcode and funct

3 register places (RT, RS, Rd), so 7-bits each

$2^7 =$ 128 possible registers

---

③ Exponent: 8-bits unsigned so $0 \rightarrow 256$

Fraction 23-bits, but since 1. Fraction, can hold 24-bit integer

so can hold $2^{24} \cdot 2^{-127}$ to $2^{24} \cdot 2^{129} = 9.86 \cdot 10^{-32}$ through $1.14 \cdot 10^{46}$

① This overflows because $(3.5 \cdot 10^{25})^2 = 1.225 \cdot 10^{51} > 1.14 \cdot 10^{46}$! By changing $3.5 \cdot 10^{25}$ to $0.00035 \cdot 10^{20}$, $(0.00035 \cdot 10^{20})^2 = 1.225 \cdot 10^{33}$, which does not overflow and the operation will succeed without problems.

② This does not over/underflow

③ This underflows because $(13.2 \cdot 10^{-20}) / (3.5 \cdot 10^{21}) = 3.77 \cdot 10^{-42} < 9.86 \cdot 10^{-32}$. Dividing $(13.2 \cdot 10^{-20}) / (0.00000035 \cdot 10^{14}) = 3.77 \cdot 10^{-32}$, which allows the operation to complete successfully