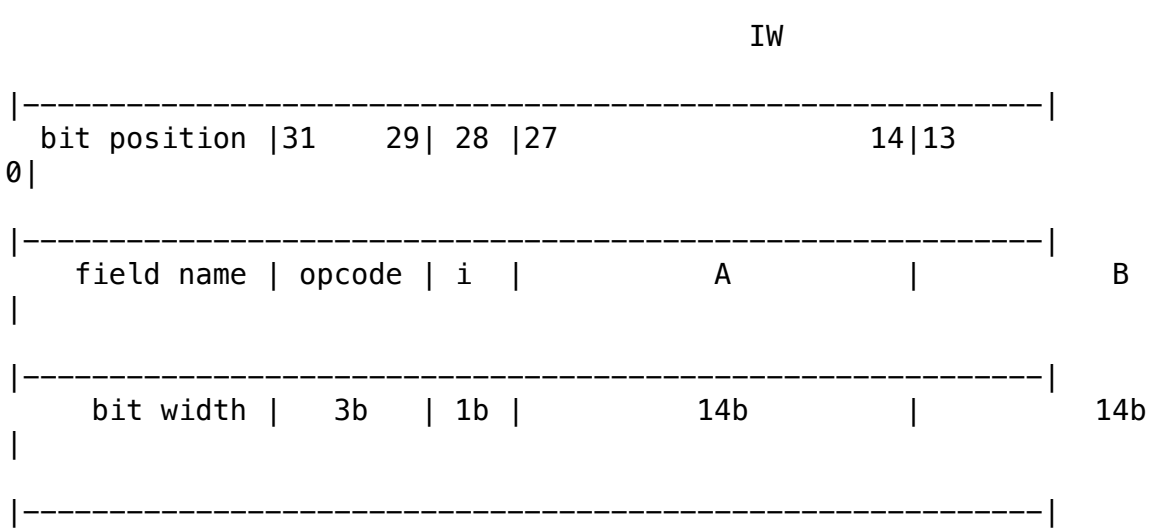
32 bit Instruction Word (IW) of SimpleCPU:



i=1 implies B is Immediate (meaning a number) as opposed to a Pointer (meaning Address)

Instruction Set of SimpleCPU:

```
ADD   -> unsigned Add
         {opcode, i} = {0, 0}
         *A <- (*A) + (*B)
         write ( readFromAddress(A) + readAddress(B) ) to address
( A )
         *A = value (content of) address A = mem[A] (mem means
memory)
         *B = value (content of) address B = mem[B]
         <- means write (assign)
         mem[A] = mem[A] + mem[B]
         mem[IW[27:14]] = mem[IW[27:14]] + mem[IW[13:0]]

ADDi  -> unsigned Add immediate
         {opcode, i} = {0, 1}
         *A <- (*A) + B
         B = read section B of the Instruction
         mem[A] = mem[A] + B
         mem[IW[27:14]] = mem[IW[27:14]] + IW[13:0]

NAND  -> bitwise NAND
         {opcode, i} = {1, 0}
         *A <- ~((*A) & (*B))

NANDi -> bitwise NAND immediate
         {opcode, i} = {1, 1}
         *A <- ~((*A) & B)

SRL   -> Shift Right if the shift amount (*B) is less than 32,
otherwise Shift Left
         {opcode, i} = {2, 0}
```

```
         *A <- ((*B) < 32) ? ((*A) >> (*B)) : ((*A) << ((*B) - 32))

SRLi  -> Shift Right if the shift amount (B) is less than 32,
otherwise Shift Left
         {opcode, i} = {2, 0}
         *A <- (B < 32) ? ((*A) >> B) : ((*A) << (B - 32))

LT    -> if *A is Less Than *B then *A is set to 1, otherwise to 0.
         {opcode, i} = {3, 0}
         *A <- ((*A) < (*B))

LTi   -> if *A is Less Than B then *A is set to 1, otherwise to 0.
         {opcode, i} = {3, 1}
         *A <- ((*A) < B)

CP    -> Copy *B to *A
         {opcode, i} = {4, 0}
         *A <- *B

CPi   -> Copy B to *A
         {opcode, i} = {4, 1}
         *A <- B

CPI   -> (regular) Copy Indirect: Copy **B to *A
         (go to address B and fetch the number then treat it as an
address and go to that address and get that data and write to
address A)
         {opcode, i} = {5, 0}
         *A <- *(*B)
         write ( readFromAddress(readFromAddress(B)) ) to address
( A )

CPIi  -> (immediate) Copy Indirect: Copy *B to **A
         (go to address B and fetch the number (*B) then go to
address A and fetch the number there and treat it as an address and
write there *B)
         {opcode, i} = {5, 1}
         *(*A) <- *B
         write ( readFromAddress(B) ) to address
( readFromAddress(A) )

BZJ   -> Branch on Zero
         (Branch to *A if *B is Zero, otherwise increment Program
Counter (PC))
         {opcode, i} = {6, 0}
         PC <- (*B == 0) ? (*A) : (PC+1)
         if(*B == 0) goTo(*A), else goTo(nextInstruction)

BZJi  -> Jump (unconditional branch)
         {opcode, i} = {6, 1}
         PC <- (*A) + B

MUL   -> unsigned Multiply
         {opcode, i} = {7, 0}
```

```
          *A <- (*A) * (*B)

MULi   -> unsigned Multiply immediate
          {opcode, i} = {7, 1}
          *A <- (*A) * B
```