

well, when I limit the 20bit immediate to 12bit, then there will be left 4 bits in 16bit to encode additional register, are there any instructions which use rd, rs1, rs2 and immediate value?



The RV32E ISA does not have any instructions that use rd, rs1, rs2, and immediate value together. However, there are some RV32E instructions that use two registers and an immediate value, and have an unused 4-bit field in their instruction encoding. Here are some examples:

ADDI: This instruction adds a 12-bit immediate value to a register and stores the result in a register. It has an unused 4-bit field in its encoding.

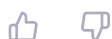
SLTI: This instruction compares a register with a 12-bit immediate value and sets a destination register to 1 if the register is less than the immediate value, or 0 otherwise. It has an unused 4-bit field in its encoding.

SLTIU: This instruction compares a register with a 12-bit unsigned immediate value and sets a destination register to 1 if the register is less than the immediate value, or 0 otherwise. It has an unused 4-bit field in its encoding.

XORI: This instruction performs a bitwise exclusive OR operation between a register and a 12-bit immediate value, and stores the result in a register. It has an unused 4-bit field in its encoding.

ORI: This instruction performs a bitwise OR operation between a register and a 12-bit immediate value, and stores the result in a register. It has an unused 4-bit field in its encoding.

You could use the unused 4-bit field in the instruction encoding to specify an additional register or other information, but this would require modifying the RV32E ISA and designing your own custom instruction set.



what are capabilities of 8bit MCUs I am considering in relation to shifting bits?, are there instructions to allow shifing by arbitrary number of bits?



Most 8-bit MCUs have shift instructions that can shift bits left or right by a fixed number of bits, typically 1 or 8 bits. However, some 8-bit MCUs also have rotate instructions that can shift bits left or right by an arbitrary number of bits. The exact capabilities of each MCU will depend on its architecture and instruction set.

For example, the AVR microcontroller family has shift instructions (LSL, LSR, ASR, ROR) that can shift bits left or right by a fixed number of bits, and also rotate instructions (ROL, ROR) that can shift bits left or right by an arbitrary number of bits.

The PIC microcontroller family also has shift instructions (ASL, ASR, RLC, RRC) that can shift bits left or right by a fixed number of bits, and rotate instructions (RLNCF, RRNCF) that can shift bits left or right by an arbitrary number of bits.

Overall, it is possible to implement bitwise operations such as shifting in software on any 8-bit MCU, even if it does not have dedicated shift or rotate instructions. However, the performance and efficiency of such software implementations will depend on the specific MCU and its clock speed.



have 8080, Z80, 6502 instructions to shift left by arbitrary number of bits?



Yes, the 8080, Z80, and 6502 microprocessors have shift instructions that can shift bits left by an arbitrary number of bits.

For the 8080, the instruction is called "DAA" (Decimal Adjust Accumulator) and can be used to shift bits left by up to 7 bits. The instruction works by shifting the contents of the accumulator register left by the specified number of bits and then performing a decimal adjustment operation on the result.

The Z80 has several shift instructions that can shift bits left by an arbitrary number of bits. The "SLA" (Shift Left Arithmetic) instruction can shift bits left by up to 7 bits, and the "SLIA" (Shift Left Immediate Arithmetic) instruction can shift bits left by up to 15 bits. Additionally,

the Z80 has a "RLC" (Rotate Left with Carry) instruction that can also be used for shifting bits left.

The 6502 has several shift instructions that can shift bits left by an arbitrary number of bits. The "ASL" (Arithmetic Shift Left) instruction can shift bits left by up to 7 bits, and the "ROL" (Rotate Left) instruction can also be used for shifting bits left by up to 7 bits.

Overall, the ability to shift bits left by an arbitrary number of bits is a common feature among many microprocessors and can be useful for implementing bitwise operations in software.



tell me details about 6502 ASL instruction



The 6502 ASL (Arithmetic Shift Left) instruction is a bitwise shift instruction that shifts the bits in a given memory location or accumulator register one position to the left, with the most significant bit (bit 7) being shifted into the carry flag and the least significant bit (bit 0) being set to 0.

The ASL instruction can be used with an accumulator operand (ASL A) or a memory operand (ASL operand), where the operand specifies the memory location to be shifted. When using a memory operand, the memory location is read from, the shift operation is performed, and the result is written back to the same memory location.

The ASL instruction updates the carry flag to reflect the value of the shifted-out bit, sets the negative flag if the result is negative (i.e., the shifted-out bit was 1), and sets the zero flag if the result is zero (i.e., all bits were shifted out).

The ASL instruction can be used to perform multiplication by powers of two, by shifting a value left by a certain number of bits to effectively multiply it by 2 to the power of that number. For example, shifting a value left by 1 bit multiplies it by 2, shifting it left by 2 bits multiplies it by 4, and so on.

The ASL instruction is a common instruction in the 6502 instruction set and is often used in conjunction with other bitwise operations to perform arithmetic and logic operations on binary data.