|  |  |
| --- | --- |
| * The array name b is like a pointer: **b == &b[0]** * **\*b == \*&b[0]** **== b[0]** * **Subtraction**: *pointer - pointer* is the number of values between the addresses. |  |

|  |  |
| --- | --- |
| **Precedences**   1. **NOT** 2. **AND/NAND** 3. **OR/XOR** 4. **IMPL** 5. **IFF** |  |

|  |  |  |
| --- | --- | --- |
| Setting a bit  Use the bitwise OR operator (|) to set a bit.  number |= 1 << x;  That will set bit x. | Clearing a bit  Use the bitwise AND operator (&) to clear a bit.  number &= ~(1 << x);  That will clear bit x. | Toggling a bit  The XOR operator (^) can be used to toggle a bit.  number ^= 1 << x;  That will toggle bit x. |
| Checking a bit  To check a bit, shift the number x to the right, then bitwise AND it:  bit = (number >> x) & 1; | Changing the (n)th bit to x  Setting the nth bit to either 1 or 0 can be achieved with the following:  number ^= (-x ^ number) & (1 << n); | Mask with Runs of (1-0-1)  0…0000 1 000 = **1 << 3**  0…0000 0111 = **(1 << 3) - 1**  0…000 111 00 = **((1 << 3) - 1) << 2**  1…111 000 11 = **~(((1 << 3) - 1) << 2)**  **~(((1 << 3) - 1) << 2)** |

**Von Neumann Architecture**

|  |  |
| --- | --- |
| (a) What are the three main parts of a von Neumann computer?  **CPU, Memory, and I/O devices**  (b) What is unique about Von Neumann architecture?  **Programs are stored as data in memory.**  (c) What are the parts of the CPU?  **Control unit and (Arithmetical/Logical) Processing Unit**  (a) What is the Memory Address Register?  **The (MAR) holds the location to read or write**  (b) What is the Memory Data Register?  **It holds the value read from memory or the value to write to memory.**  (c) What are the steps involved in reading and writing memory?  **To read: Set MAR to the address to read;**  **signal Read;**  **Find the value in the MDR and copy it out to wherever.**  **To write: Set (MAR) to the address to read;**  **Set MDR ← value to write;**  **Signal Write**  What does the Program Counter point to? Why is it badly named?  **The PC points to the next instruction to execute.**  What are the three basic types of instructions?  **Calculation instructions** use and create values;  **Data movement instructions** move data to/from memory;  **Control instructions** perform goto operations. | (a) What are the phases of the instruction cycle?  **Fetch Instruction, Decode Instruction, Evaluate Addresses, Fetch**  **Operands, Execute Instruction, Store Results.**  What are the steps of the Fetch Instruction phase?  **Read the instruction pointed to the program counter from memory into the instruction register; then increment the program counter**  During the Decode Instruction phase of the instruction cycle, where is the instruction being decoded?  **The instruction is in the instruction register.**  What happens during the Evaluate Addresses (a.k.a Get Effective  Addresses) phase of the instruction cycle? Why is this phase sometimes skipped?  **We figure out where the operands are stored; this could be a register number or a memory address.**  What happens during the Fetch Operands phase of the instruction cycle? From where can operands be fetched? **During Fetch Operands we actually retrieve the operand values from a register, from memory, or from part of the instruction register.**  (f) What happens during the Store Results phase of the instruction cycle? To where can results be stored? Why is the phase sometimes skipped?  **During Store Results, values calculated during Execute Instruction are moved to memory or CPU registers. This phase is skipped if there are no instructions.** |