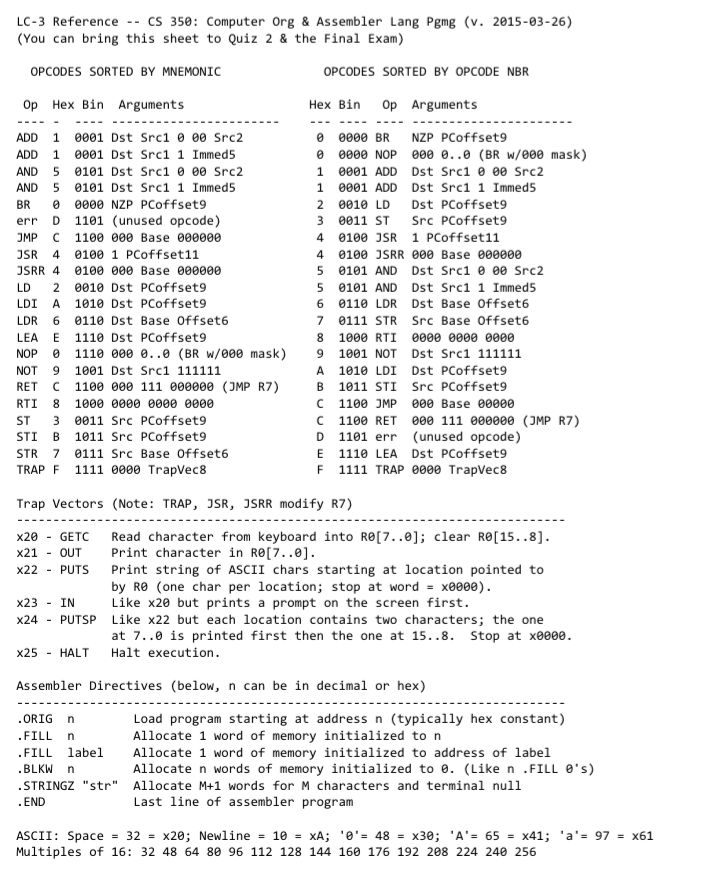
|  |  |  |
| --- | --- | --- |
| **Setting a bit**  Use the bitwise OR operator (|) to set a bit.  number |= 1 << x;  That will set bit x. | **Clearing a bit**  Use the bitwise AND operator (&) to clear a bit.  number &= ~(1 << x);  That will clear bit x. | **Toggling a bit**  The XOR operator can be 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;  That will put the value of bit x into the variable bit. | **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);  Bit n will be set if x is 1, and cleared if x is 0. | **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)** |

|  |  |
| --- | --- |
| * 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. | **ADD set\_cc(cpu,cpu->reg[dst]);**  1 cpu->reg[dst] = cpu->reg[src1]+imm5;  0 cpu->reg[dst] = cpu->reg[src1]+cpu->reg[src2];  **AND (set\_cc)**  1 cpu->reg[dst] = cpu->reg[src1]&cpu->reg[src2];  0 cpu->reg[dst] = cpu->reg[src1]&imm5;  **LD (set\_cc):** cpu->reg[dst] = cpu->mem[cpu->pc+pcoffset];  **LDI (set\_cc):** cpu->reg[dst] = cpu->mem[cpu->mem[cpu->pc+pcoffset]];  **LDR (set\_cc):** cpu->reg[dst] = cpu->mem[cpu->reg[base]+offset];  **LEA (set\_cc):** cpu->reg[dst] = cpu->pc+pcoffset;  **BR:** if(mask&get\_cc(cpu)!=0){cpu->pc += pcoffset;}  **JMP:** cpu->pc = cpu->reg[base];  **JSR:** 1 cpu->pc = cpu->pc+pcoffset;  **JSRR**: 0 cpu->pc = cpu->reg[base];  **NOT (set\_cc):** cpu->reg[dst] = ~cpu->reg[src];  **ST:** cpu->mem[cpu->pc+pcoffset] = cpu->reg[src];  **STI:** cpu->mem[cpu->mem[pcoffset+cpu->pc]] = cpu->reg[src];  **STR:** cpu->mem[base+offset] = cpu->reg[dst];  **TRAP**  **case 32**: cpu->reg[0] = prompt\_char();  **case 33**:printf("%c",cpu->reg[0]);  **case 34**:tmp\_addr = cpu->reg[0];  while(cpu->mem[tmp\_addr]!=0){  printf("%c",cpu->mem[tmp\_addr]);  tmp\_addr++;}  **case 35**:cpu->reg[0] = prompt\_char();  printf("char read: '%c'\n",cpu->reg[0]);  **case 37**:cpu->running = 0;  set\_cc(cpu,1);  **default**:set\_cc(cpu,cpu->reg[7]); |

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

****