Addressing Modes

The 6502 processor provides several ways in which memory locations can be addressed. Some instructions support several different modes while others may only support one. In addition the two index registers can not always be used interchangeably. This lack of orthogonality in the instruction set is one of the features that makes the 6502 trickier to program well.

Implicit

For many 6502 instructions the source and destination of the information to be manipulated is implied directly by the function of the instruction itself and no further operand needs to be specified. Operations like 'Clear Carry Flag' (CLC) and 'Return from Subroutine' (RTS) are implicit.

Accumulator

Some instructions have an option to operate directly upon the accumulator. The programmer specifies this by using a special operand value, 'A'. For example:

```
        LSR A           ;Logical shift right one bit
        ROR A           ;Rotate right one bit
```

Immediate

Immediate addressing allows the programmer to directly specify an 8 bit constant within the instruction. It is indicated by a '#' symbol followed by an numeric expression. For example:

```
        LDA #10         ;Load 10 ($0A) into the accumulator
        LDX #LO LABEL   ;Load the LSB of a 16 bit address into X
        LDY #HI LABEL   ;Load the MSB of a 16 bit address into Y
```

Zero Page

An instruction using zero page addressing mode has only an 8 bit address operand. This limits it to addressing only the first 256 bytes of memory (e.g. $0000 to $00FF) where the most significant byte of the address is always zero. In zero page mode only the least significant byte of the address is held in the instruction making it shorter by one byte (important for space saving) and one less memory fetch during execution (important for speed).

An assembler will automatically select zero page addressing mode if the operand evaluates to a zero page address and the instruction supports the mode (not all do).

```
        LDA $00         ;Load accumulator from $00
        ASL ANSWER      ;Shift labelled location ANSWER left
```

Zero Page,X

The address to be accessed by an instruction using indexed zero page addressing is calculated by taking the 8 bit zero page address from the instruction and adding the current value of the X register to it. For example if the X register contains $0F and the instruction LDA $80,X is executed then the accumulator will be loaded from $008F (e.g. $80 + $0F => $8F).

NB:
The address calculation wraps around if the sum of the base address and the register exceed $FF. If we repeat the last example but with $FF in the X register then the accumulator will be loaded from $007F (e.g. $80 + $FF => $7F) and not $017F.

```
        STY $10,X       ;Save the Y register at location on zero page
        AND TEMP,X      ;Logical AND accumulator with a zero page value
```

Zero Page,Y

The address to be accessed by an instruction using indexed zero page addressing is calculated by taking the 8 bit zero page address from the instruction and adding the current value of the Y register to it. This mode can only be used with the LDX and STX instructions.

```
        LDX $10,Y       ;Load the X register from a location on zero page
        STX TEMP,Y      ;Store the X register in a location on zero page
```

Relative

Relative addressing mode is used by branch instructions (e.g. BEQ, BNE, etc.) which contain a signed 8 bit relative offset (e.g. -128 to +127) which is added to program counter if the condition is true. As the program counter itself is incremented during instruction execution by two the effective address range for the target instruction must be with -126 to +129 bytes of the branch.

```
        BEQ LABEL       ;Branch if zero flag set to LABEL
        BNE *+4         ;Skip over the following 2 byte instruction
```

Absolute

Instructions using absolute addressing contain a full 16 bit address to identify the target location.

```
        JMP $1234       ;Jump to location $1234
        JSR WIBBLE      ;Call subroutine WIBBLE
```

Absolute,X

The address to be accessed by an instruction using X register indexed absolute addressing is computed by taking the 16 bit address from the instruction and added the contents of the X register. For example if X contains $92 then an STA $2000,X instruction will store the accumulator at $2092 (e.g. $2000 + $92).

```
        STA $3000,X     ;Store accumulator between $3000 and $30FF
        ROR CRC,X       ;Rotate right one bit
```

Absolute,Y

The Y register indexed absolute addressing mode is the same as the previous mode only with the contents of the Y register added to the 16 bit address from the instruction.

```
        AND $4000,Y     ;Perform a logical AND with a byte of memory
        STA MEM,Y       ;Store accumulator in memory
```

Indirect

JMP is the only 6502 instruction to support indirection. The instruction contains a 16 bit address which identifies the location of the least significant byte of another 16 bit memory address which is the real target of the instruction.

For example if location $0120 contains $FC and location $0121 contains $BA then the instruction JMP ($0120) will cause the next instruction execution to occur at $BAFC (e.g. the contents of $0120 and $0121).

```
        JMP ($FFFC)     ;Force a power on reset
        JMP (TARGET)    ;Jump via a labelled memory area
```

Indexed Indirect

Indexed indirect addressing is normally used in conjunction with a table of address held on zero page. The address of the table is taken from the instruction and the X register added to it (with zero page wrap around) to give the location of the least significant byte of the target address.

```
        LDA ($40,X)     ;Load a byte indirectly from memory
        STA (MEM,X)     ;Store accumulator indirectly into memory
```

Indirect Indexed

Indirect indirect addressing is the most common indirection mode used on the 6502. In instruction contains the zero page location of the least significant byte of 16 bit address. The Y register is dynamically added to this value to generated the actual target address for operation.

```
        LDA ($40),Y     ;Load a byte indirectly from memory
        STA (DST),Y     ;Store accumulator indirectly into memory
```

[<< Back] [Home] [Contents] [Next >>]