6502 ADDRESSING MODES

As with most processors, the 6502 has its own variations on the standard addressing modes. Certain instructions may refer to memory using some or all of the available modes. Some can only use one of the addressing modes.

Absolute aaaa

The operand is a 2 byte memory address.

LDA	$3491	Load the accumulator with the contents of memory location $3491 (13457 decimal).
LDA	num1	Load the accumulator with the contents of memory location labelled num1.

Zero Page aa

The operand is a 1 byte memory address. This means that it must lie in the first 256 bytes (zero page) of memory ($0000 to $00FF).

LDA	$56	Load the accumulator with the contents of memory location $0056.

This makes the instruction one byte shorter than its absolute addressing equivalent. This saves memory space, and also means that the instruction is executed faster, which may be important.

LDA	num2	An assembler will automatically generate a zero page instruction code if the label refers to a zero page address.

Immediate #aa

The operand is a 1 byte data item (not an address). It is indicated by a hash symbol (#).

LDA	#48	Load the accumulator with the value $48 (which happens to be the ASCII code for H.)
LDA	#'H'	By using single quotes, the assembler will let you use the character itself.

Implicit

Some instructions don't require an operand, as the location of the data to be manipulated is implied by the instruction itself. This means that the assembler can save memory by just storing the instruction code in memory.

CLC	Clear the carry bit in the flags register.
TAX	Transfer the contents of the accumulator to index register X.

Accumulator A

Some instructions (mainly the shift and rotate group) can be applied to data held in memory locations and also to data held in the accumulator. You indicate the latter case by using A as your operand.

ASL	A	Shift the contents of the accumulator one bit to the left.
ROR	A	Rotate the contents of the accumulator one bit to the right.

Absolute Indexed, X aaaa, X
Absolute Indexed, Y aaaa, Y

The operand is a 2 byte memory address. The location of the data is calculated by taking this address, and adding to it the contents of the index register.

```
LDX   #$0A      Load the X register with $0A (10 decimal)
LDA   $2020, X  Load the accumulator with contents of memory location $202A ($2020 + $0A).
```

Typically used to access tables of data items, where $2020 is the base address, and the value in X (or Y) is the offset within the table.

Zero Page Indexed, X aa, X
Zero Page Indexed, Y aa, Y

Similar to the Absolute Indexed modes above, except that the operand is a 1 byte memory address in the zero page of memory.

```
LDY   #$4A      Load the Y register, with $4A.
LDA   $24, Y    Load the accumulator with the contents of memory location $6E ($4A + $24).
```

Indirect Absolute (aaaa)

Only one 6502 instruction uses the indirect addressing mode in its simplest form. The JMP instruction. The operand is a 2 byte address, which contains the least significant byte of the target address. The most significant byte of the target address is in the memory location following . Transfer is controlled to the instruction in the location referred to by the target address.

```
If memory location   $3418   contains   $55
                     $3419   contains   $29

JMP   ($3418)   Will cause the processor to jump to the instruction in location $2955.
```

Indexed Indirect (aa, X)

This is a mixture of indexed addressing and indirect addressing, which can only be used with the X register. It can be summarised as ‘Add the offset and then find the address’.

The operand is a 1 byte zero page address. The contents of the X register are added to it, and the resulting location will contain the least significant byte of a 2 byte address, which contains the data. For example:

```
LDX   #$11
LDA   ($36, X)
```

will mean that the resulting zero page address is $47 ($36 + $11). If location $0047 contains $49, and location $0048 contains $A3, then the target address is $A349. The contents of location $A349 will be loaded into the accumulator.

This mode is often used to access a table of addresses held on the zero page, where the base address of the table is supplied as the operand, and the offset within the table is held in the index register X.

Indirect Indexed (aa), Y

This is also a mixture of indexed and indirect addressing, but this one can only be used with the Y register. It can be summarised as 'Find the address and then add the offset'.

The operand is a 1 byte zero page address, which contains the least significant byte of a 2 byte address. The most significant byte is held in the next byte (aa+1). To that 2 byte address, add the contents of the Y register. The resulting address contains the data. For example, if we assume that

location	$A5	contains	$28
	$A6	contains	$77

```
LDY  $08
LDA  ($A5), Y
```

will result in the address $7728 having $08 added to it to produce the address $7730. The content of location $7730 will then be loaded into the accumulator.

Relative aa

Relative addressing is only used with branch instructions.

The operand is a 1 byte 2's complement number in the range -128 to +127. This is added to the address of the instruction following the branch instruction, to give the address of the instruction to be jumped to.

```
BEQ  #$F5      F5 is equivalent to -11 decimal.
```

This will cause the processor to take the address of the instruction immediately following the branch instruction, subtract 11 from it, and carry on from whichever instruction is at that location – provided that the zero flag is set.

Luckily, the assembler allows you to use labels, so that you don't have to keep counting bytes.

```
LB1:  LDA    $FA55
      INX
      BEQ    LB1
```