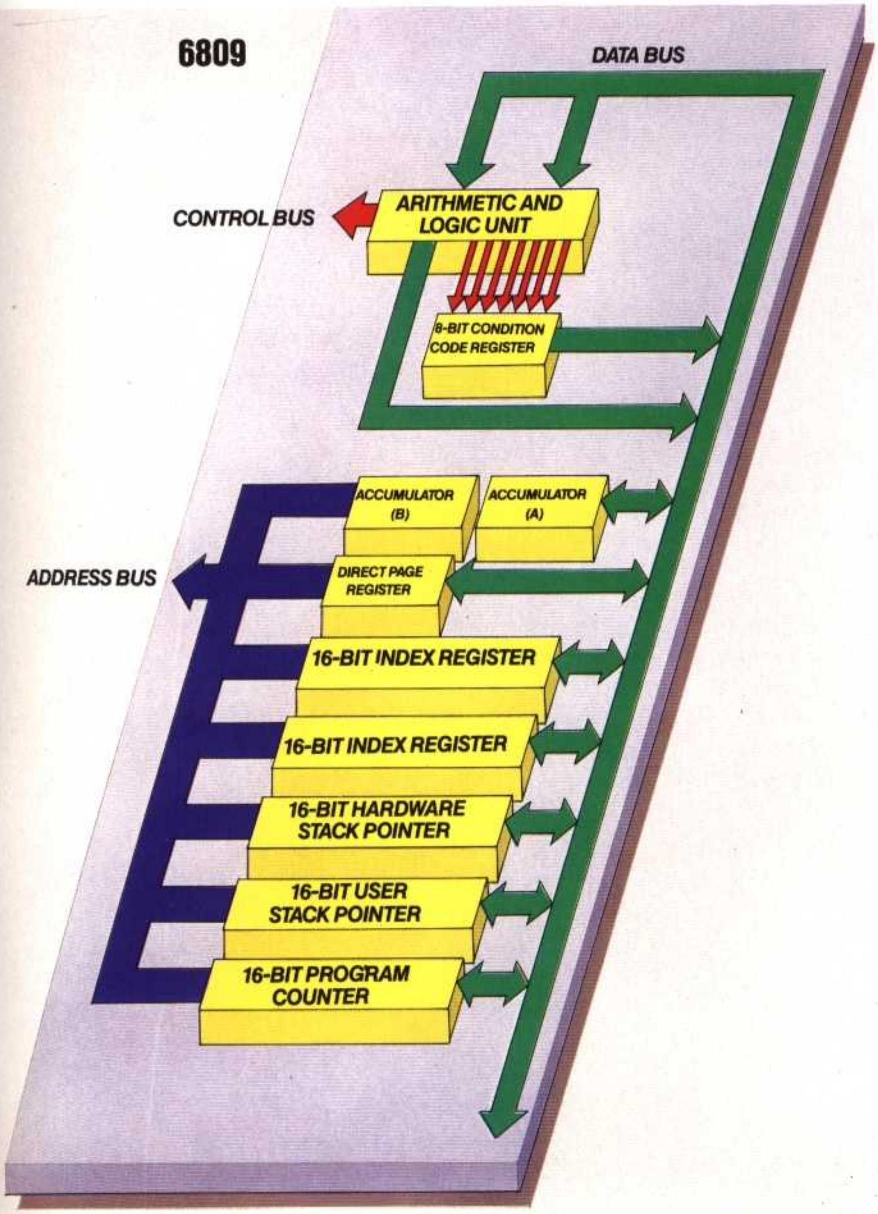
'Assembler directives' is a better name as this explains their function — to direct the functioning of the Assembler program. If these directives are labelled, then the label will be translated by the Assembler into the appropriate address — so we may have, for example:

NUM1 FCB 0	reserving a single byte that will be referred to as NUM1, with initial value 0
NUM2 FCB 0	similar to the above
NUM3 FDB #A93B	reserves two bytes for the 16-bit number #A93B (#, the 'hash' sign, is often used by 6809 Assemblers as a sign that the number is in hexadecimal notation)

The following instructions load the values stored in these locations into various registers:

LDA NUM1	will load the eight-bit number stored at the memory location represented by NUM1 into accumulator A
LDB NUM2	as above, loads NUM2 into accumulator B
LDX NUM3	
LDY NUM3	These instructions will load the 16-bit
LDS NUM3	number in NUM3 into the X, Y, S, U and
LDU NUM3	D registers respectively
LDD NUM3	



KEVIN JONES

In a similar way, the eight- or 16-bit contents of a register may be stored in a memory location by using one of:

```
STA NUM1
STB NUM2
STX NUM3
STY NUM3
STS NUM3
STU NUM3
STD NUM3
```

Notice that when the accumulator is loaded from NUM1, you actually copy NUM1 into the accumulator without changing it; the store operations function similarly.

The contents of two registers may be exchanged (provided that they are the same size) by using the EXG instruction. For example:

EXG A,B	exchanges the contents of registers A and B
EXG X,S	exchanges the contents of registers X and S

The contents of one may be transferred to another — for example: TFR Y,U copies the contents of Y into U. To accomplish this, the two registers must again be of the same size, both eight-bit or both 16-bit.

In order to write a program that actually does something, let us introduce the ADD instruction, which will add the contents of a memory location to the contents of one of the accumulators. It takes the form:

ADDA NUM1	meaning 'add the contents of memory location NUM1 into the A register, leaving the A register containing the result of the addition'

First we will add the two eight-bit numbers in NUM1 and NUM2, putting the answer back in NUM1 and ignoring any overflow if their sum is larger than an eight-bit number. We will then add the two locations' contents again, but this time obtaining a 16-bit result in NUM3.

First example:

LDA NUM1	copy first number into A
ADDA NUM2	add second number
STA NUM1	store answer back in NUM1

Second example:

LDB NUM1	copy first number into B
SEX	convert the eight-bit number in B into a 16-bit number in D
STD NUM3	copy D into NUM3
LDB NUM2	copy second number into B
SEX	convert it to 16-bit number in D
ADDD NUM3	add the first 16-bit number from NUM3 into D
STD NUM3	store the answer back in NUM3