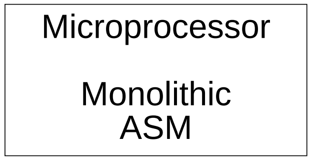
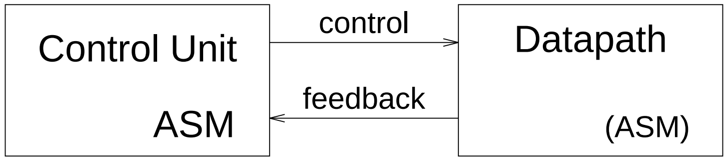
# Microprocessor Internals

- Naive view

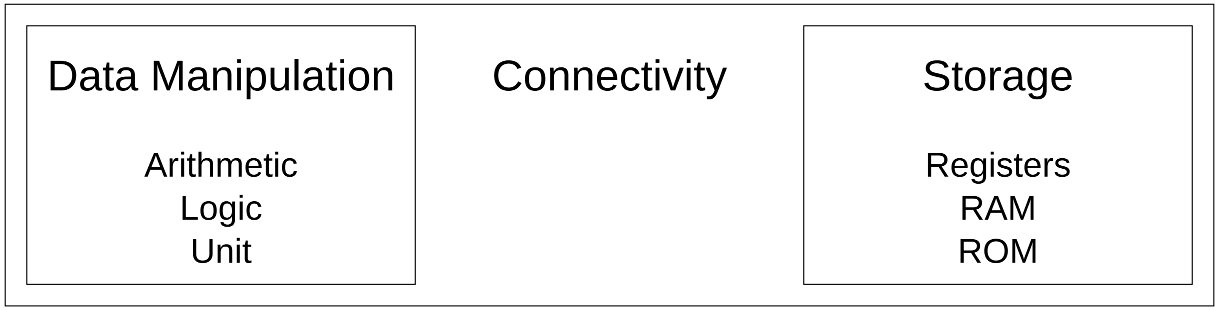



- Linked state machines



# Microprocessor Internals

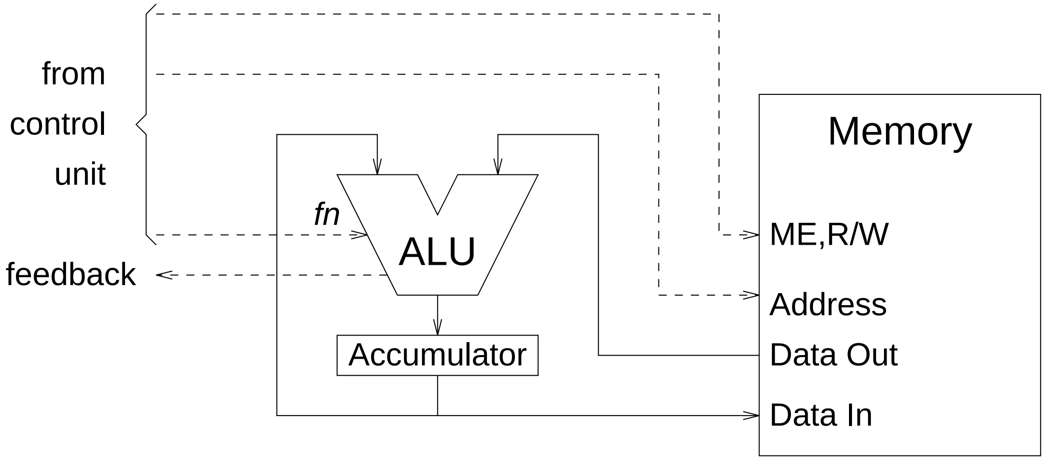
- Datapath




– Top Down Design
  - - How many registers?
  - - How many busses?
  - - What data manipulation functions?

# Datapath

- Simple Datapath




  – Register Memory Architecture – Single Address Architecture
  Each ALU instruction is of the form;

$$Acc' \leftarrow \mathcal{F}\{Acc, mem(address)\}.$$

  Since the accumulator is used as operand and destination, only one address need be specified.

# Expression Evaluation

- Consider the HLL statement:

```
W := (X + Y) * Z
```

- Evaluation:

| Instruction | RTL description |
|---|---|
| `LDA x` | $Acc' \leftarrow mem(x)$ |
| `ADD y` | $Acc' \leftarrow Acc + mem(y)$ |
| `MUL z` | $Acc' \leftarrow Acc \times mem(z)$ |
| `STA w` | $mem(w)' \leftarrow Acc$ |

where $W = mem(w), X = mem(x), Y = mem(y), Z = mem(z)$.
i.e. $W$ is a variable stored in the memory at location $w \ldots$

# ALU Functionality

ALU performs a number of unary and binary arithmetic and logical functions:

- Unary functions
  - A `NOP`
  - ~A `COM`
  - $A << 1$ `LSL`
  - $A >> 1$ `LSR`
  - -A `NEG`
  - $A + 1$ `INC`
  - $A - 1$ `DEC`
  - M `LDA m`

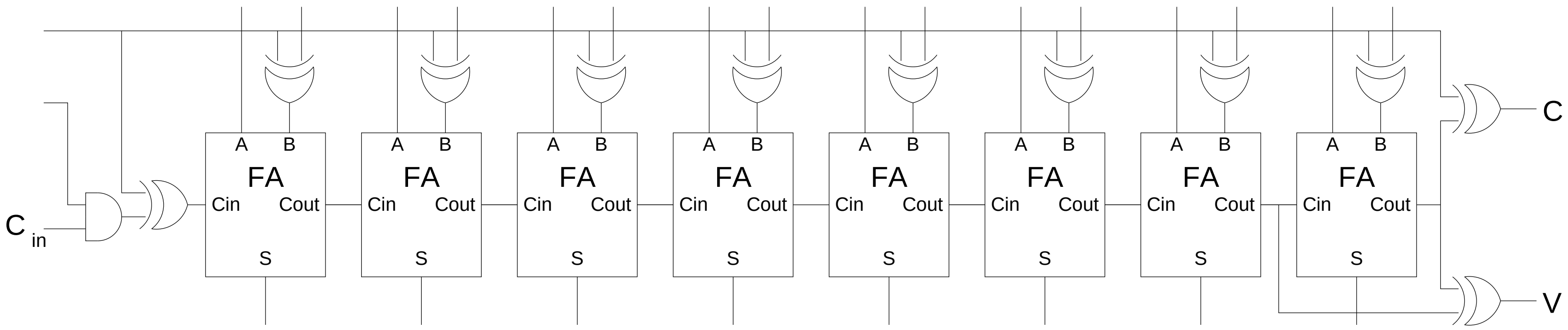
# ALU Functionality

- Binary functions
  - $A \& M$ `AND m`
  - $A \mid M$ `OR m`
  - $A$ ^ $M$ `EOR m`
  - $A + M$ `ADD m`
  - $A - M$ `SUB m`

- Others
  - $\emptyset$ `CLR`

# ALU Design

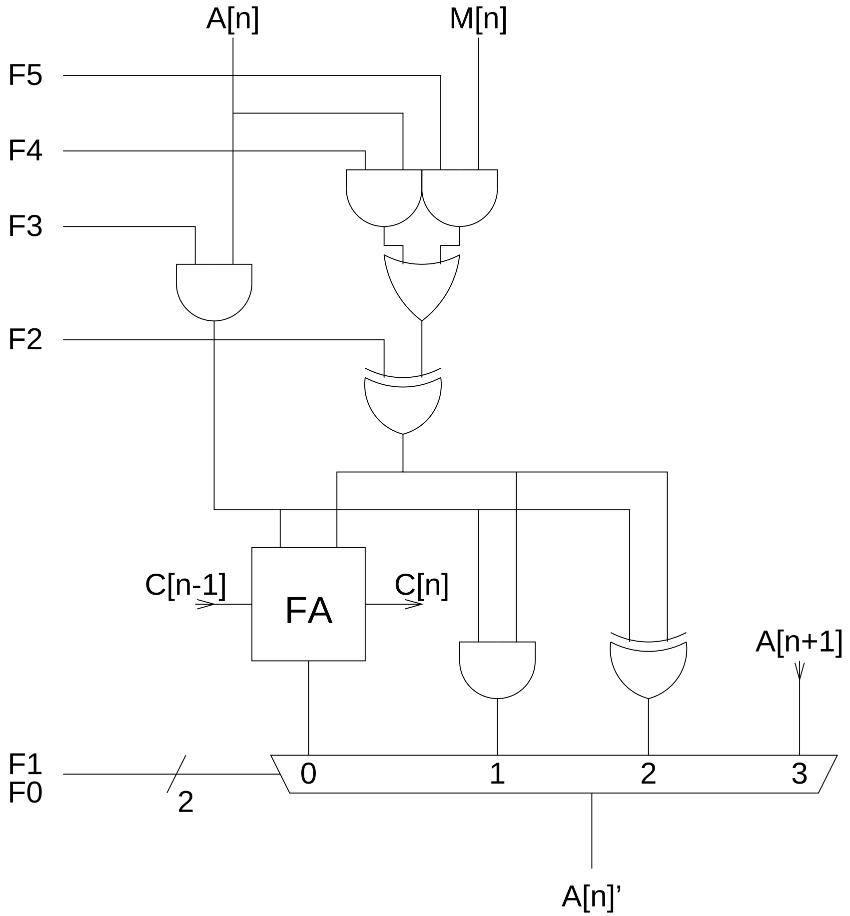
The ALU is designed around the integer adder unit:




| | | | |
|---|---|---|---|
| A, | `NOP`, | is implemented as | $\emptyset$ + A |
| A << 1, | `LSL`, | is implemented as | A + A |
| -A, | `NEG`, | is implemented as | $\emptyset$ - A |
| M, | `LDA m`, | is implemented as | $\emptyset$ + M |
| $\emptyset$, | `CLR`, | is implemented as | A - A |

# ALU Design

A little additional logic completes the ALU functionality[1].




[1]note that two of the additional gates may exist within the full adder.

# ALU Functionality

- Expensive Functions

  The instruction set so far has been based on the arithmetic/logical instructions of the MC6809 processor.

  The following functions are expensive in terms of ALU area and have been initially omitted from our design:

  - A * M `MUL m`

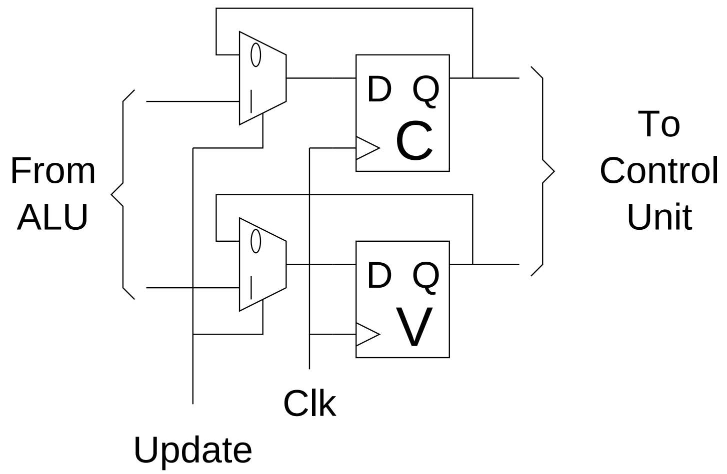
    requires $n - 1$ $n$-bit adders for an $n$-bit multiplication

  - A << M `LSL m`
  - A >> M `LSR m`

    require a barrel shifter

# ALU Feedback

- Status Flags exist to retain overflow information between arithmetic operations:




  - the latched overflow information is fed back to the controller.
  - the flags are only updated when an arithmetic operation is performed.

## Multi-word Arithmetic

- The carry out overflow flag can be used to support multi-word signed and unsigned arithmetic:

  Evaluation of `Z := X + Y` for double-word addition[2]:

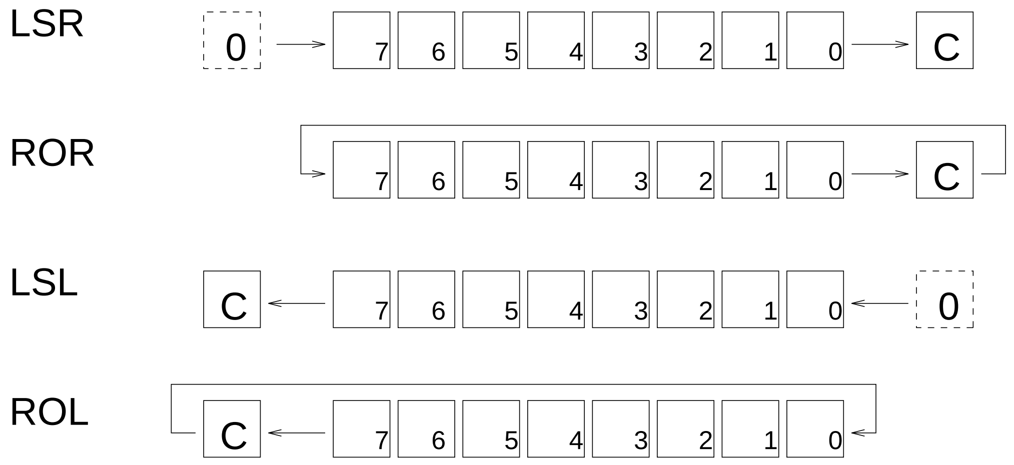
| | |
|---|---|
| `LDA x_lo` | $Acc' \leftarrow mem(x\_lo)$ |
| `ADD y_lo` | $\{C' : Acc'\} \leftarrow Acc + mem(y\_lo)$ |
| `STA z_lo` | $mem(z\_lo)' \leftarrow Acc$ |
| `LDA x_hi` | $Acc' \leftarrow mem(x\_hi)$ |
| `ADC y_hi` | $\{C' : Acc'\} \leftarrow Acc + mem(y\_hi) + C$ |
| `STA z_hi` | $mem(z\_hi)' \leftarrow Acc$ |

  - after the ADD instruction the intermediate carry is stored in $C$.
  - for the ADC instruction the ALU is configured such that $C$ is used for carry in and carry out.

[2]use SUB and SBC for subtraction

# Multi-word Arithmetic

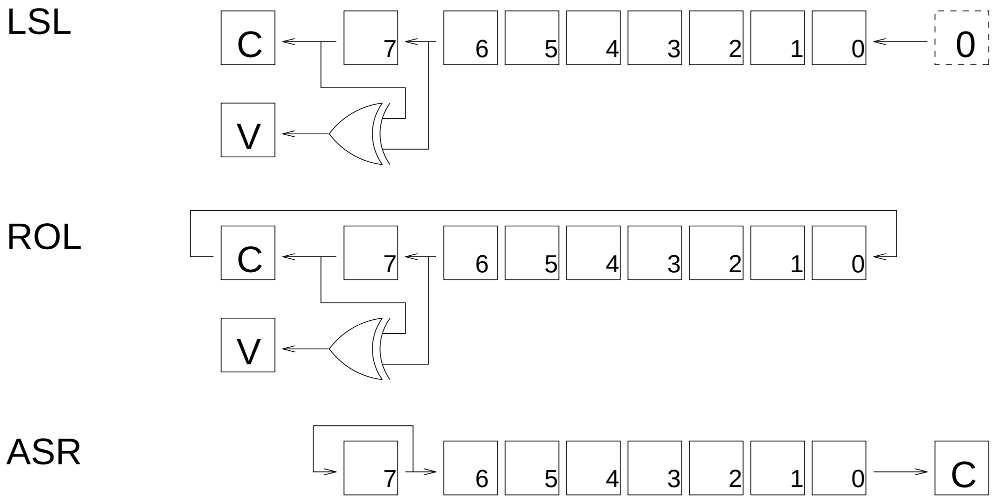
- Multi-word logical shifts:




  - `LDA x_hi; LSR; STA x_hi; LDA x_lo; ROR; STA x_lo;` performs double word shift right.
  - `LDA x_lo; LSL; STA x_lo; LDA x_hi; ROL; STA x_hi;` performs double word shift left.

# More Shifts

- Shifts of signed numbers (arithmetic shifts):



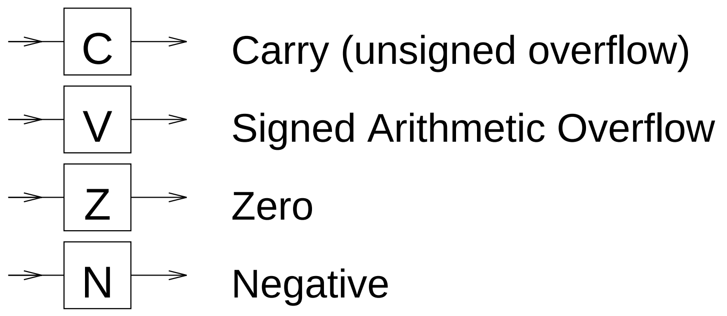

  – `LSL` and `ROL` are used for multiplication by 2. These instructions must set $V$ in the case of a signed arithmetic overflow.
  – A special instruction `ASR`[3] performs divide by 2 for signed numbers.

[3] `LSL` is also known as `ASL`

# ALU Feedback

Other signals, $N$ and $Z$, are fed back to the controller to indicate whether a result is negative or zero.

These signals may be taken directly from the accumulator or they may be latched, like $C$ and $V$, only on appropriate instructions[4].




The four feedback signals are used to control the program flow.

[4]flags are usually latched for processors with more than one data register