The Assembly Language

Lesson 2 – The 68000

Motorola 68000 structure

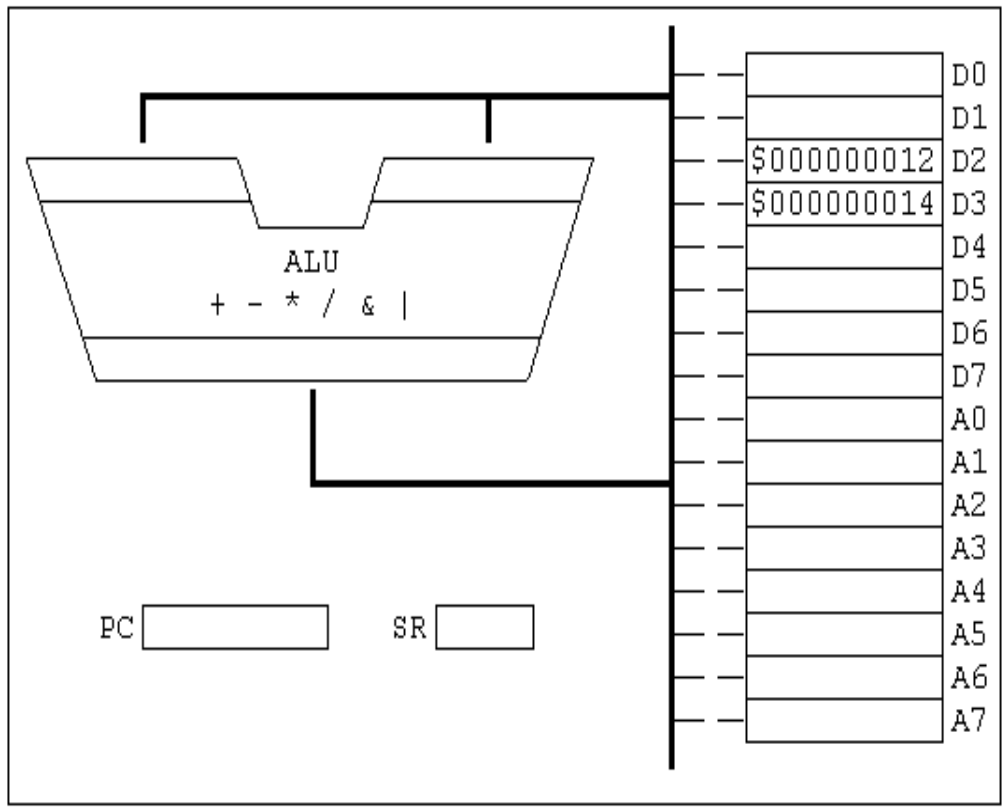
- ALU
- 16 registers:
  - 8 Data Registers (D0..D7)
  - 8 Address Registers (A0..A7)
- PC (Program Counter)
- SR (Status Register)

How it works

- The following example will shoe how the ALU and registers and connected, and which are the steps of each single operation.
- The instruction we will study is
  "ADD.L D2, D3"
- The instruction will add the content of D2 to the content of D3

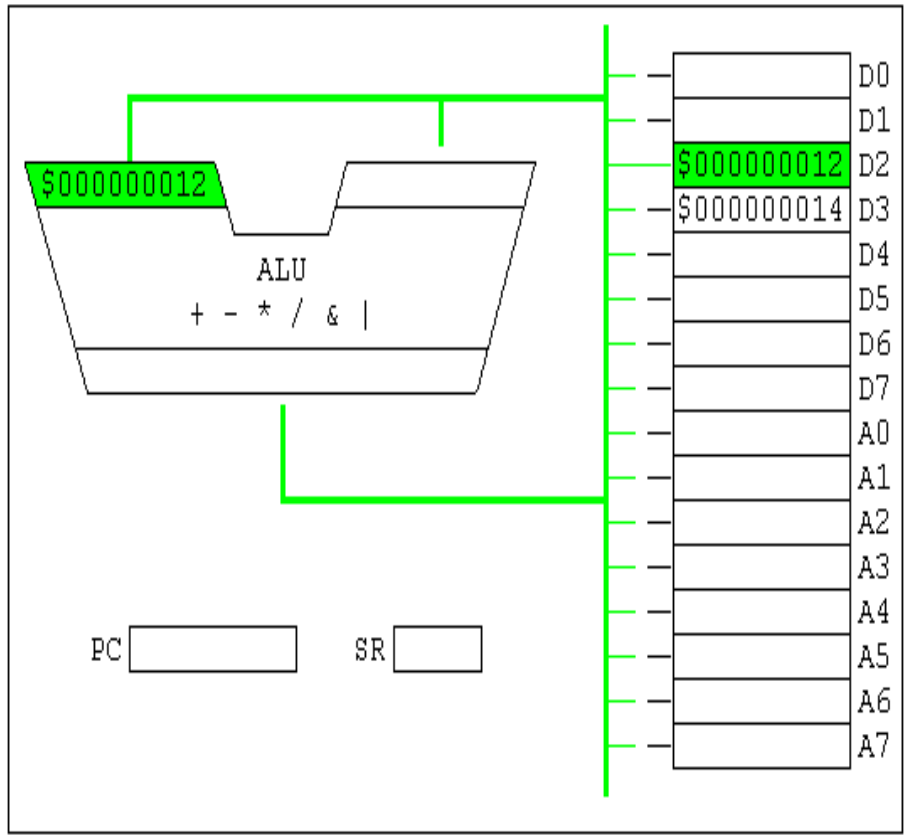
Initial processor state

- Initially the internal bus is in idle mode
- We have the values $12, $14 in the registers D2, D3



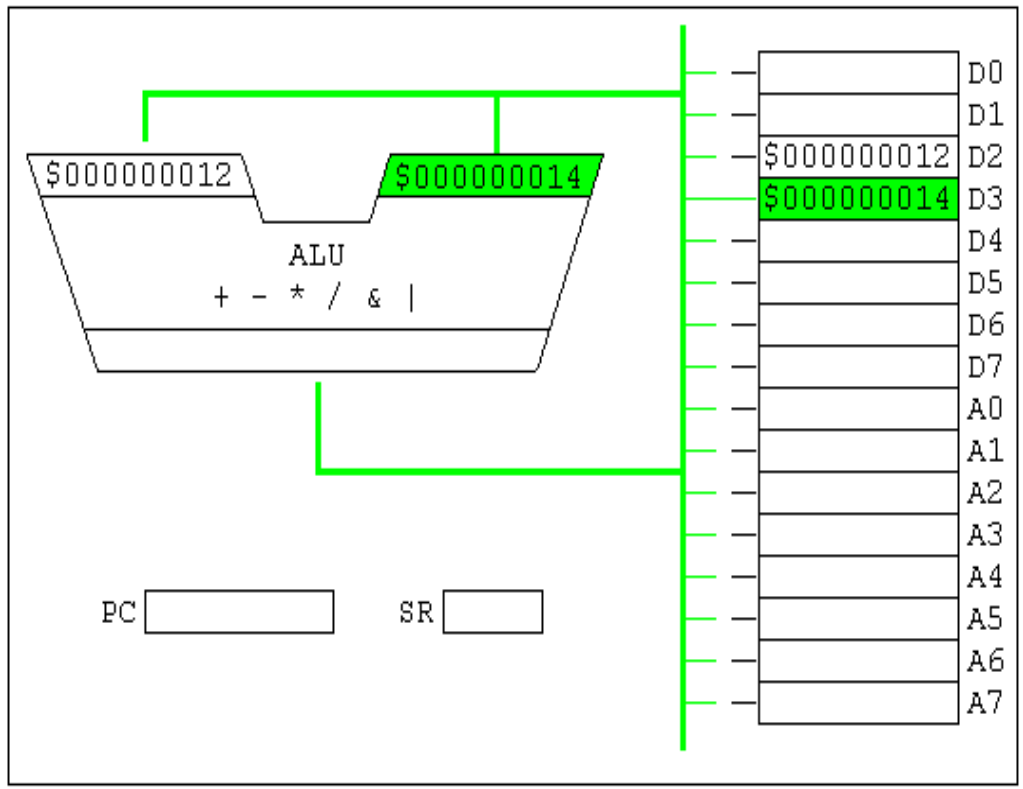
Step 1

- The first data (D2) is loaded in the ALU
- A link between the ALU and the bus is established as well as a link between the bus and D2



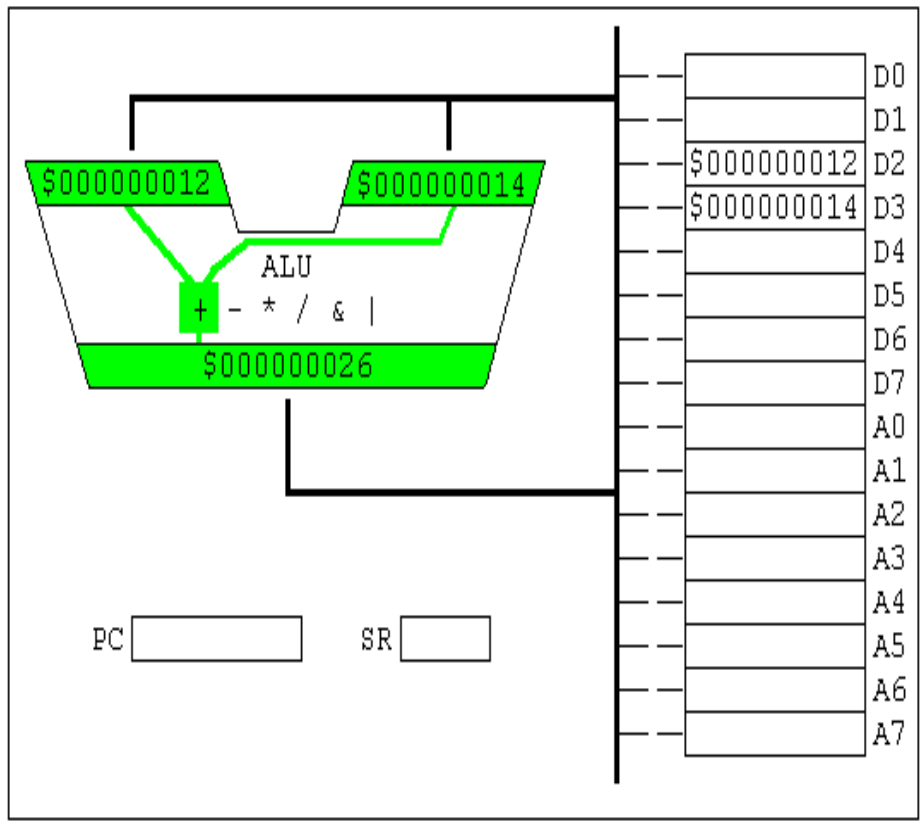
Step 2

- We now load D3



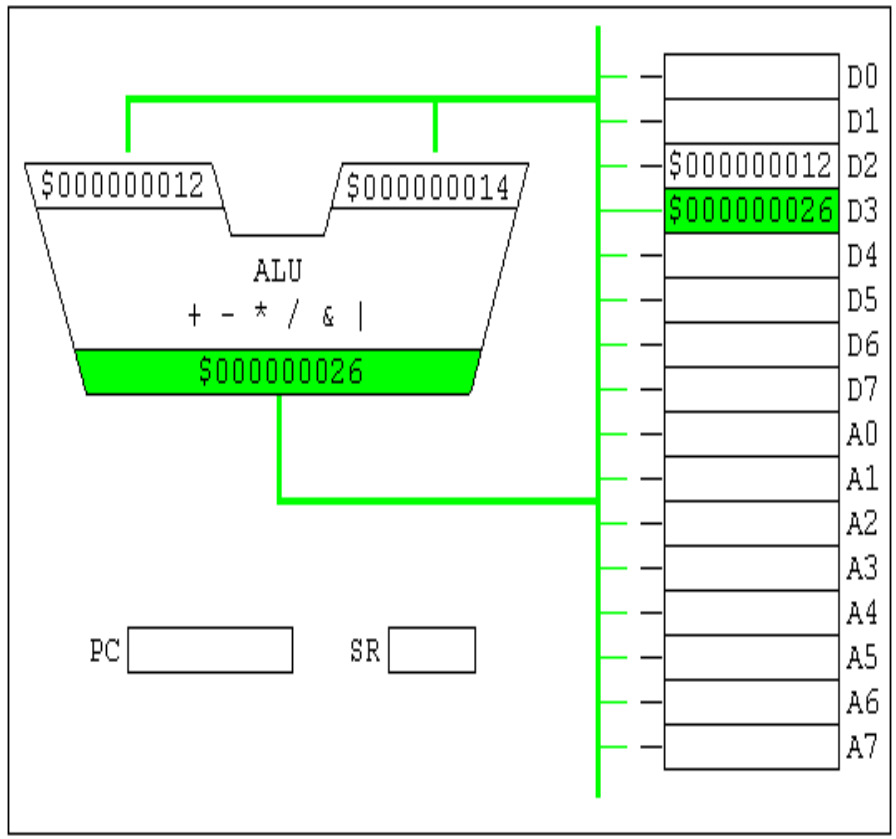
Step 3

- We now have to start the addition
- The internal processor bus is no longer used
- We activate the internal ALU bus



Step 4

- We now copy the result into the register D3
- We have to re-create a link



How to add numbers in memory

- We DO NOT add directly from memory
- We load the first value in a register
- Load the second value in another register
- Add both numbers from the registers
- Copy the register containing the result to memory

What are the registers?

- Some small micro switches which can be set to anything we want
- 5 sorts of registers:
  - Data Registers D0..D7: 32bits
  - Address Registers A0..A6: 32 bits
  - Stack Register A7: 32bits
  - Status Register: 16bits
  - Program Counter: 24bits

A deeper view of the SR

- Divided in 2 parts
- A systembyte (bits 15-8)
- A userbyte, or flagregister (bits 7-0)

Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

| T | - | S | - | - |I2,1,0| -| -| -|X|N|Z|V|C

The systembyte

- Bit 15: T – The trace bit
  - if set, an interrupt will be called after each instruction, used for debugging
- Bit 13: S – Supervisor bit
  - If set, you get “more” access to some instructions. Do not use it unless you want to program an OS
- Bit 8-10: Interrupt mask
  - Set an interrupt level to give different priority

The flagregister

- C-flag (Carry)
- V-flag (oVerflow)
- Z-flag (Zero)
- N-flag (Negative)
- X-flag (eXtended)

Some information about instructions

- Use a suffix to specify the size of the operation
  - .B : 8 bits
  - .W : 16 bits
  - .L : 32 bits
- Example: MOVE.W D0, D1 moves the lower 16 bits from D0 to D1

Effective Address

- The are 14 ways to address things in memory
- We will use the example of the MOVE instruction

Immediate Addressing With Data Registers

- Syntax: Dn (n is 0-7)
- Example: Move.L D1, D0

Instruction	Before	After
Move.B D1, D0	D0=FFFFFFFF D1= 01234567	D0=FFFFFF67 D1=01234567
Move.W D1, D0	D0=FFFFFFFF D1=01234567	D0=FFFF4567 D1=01234567

Immediate Addressing With Address Registers

- Syntax: An (n is 0-7)
- Example: Move.L A1, D0

Instruction	Before	After
Move.W A1, D0	D0=FFFFFFFF A1= 01234567	D0=FFFF4567 A1=01234567
Move.W D0, A1	D0=01234567 A1=FFFFFFFF	D0=01234567 D1=00004567
Move.W D0, A1	D0=0000FFFF A1=00000000	D0=0000FFFF A1=FFFFFFFF

Indirect Addressing With Address Registers

- Syntax: (An) (n is 0-7)
- Example: Move.L (A1), D0

Instruction	Before	After
Move.L (A1), D0	D0=FFFFFFFF A1= 00001000 $1000=01234567	D0=01234567 A1=00001000 $1000=01234567

Indirect Addressing With Address Registers with Afterincreasment

- Syntax: (An)+ (n is 0-7)
- Example: Move.L (A1)+, D0

Instruction	Before	After
Move.L (A1)+, D0	D0=FFFFFFFF A1= 00001000 $1000=01234567	D0=01234567 A1=00001004 $1000=01234567

Indirect Addressing With Address Registers with Predecrement

- Syntax: -(An) (n is 0-7)
- Example: Move.L -(A1), D0

Instruction	Before	After
Move.L -(A1), D0	D0=FFFFFFFF A1= 00001004 $1000=01234567	D0=01234567 A1=00001000 $1000=01234567

Indirect Addressing With Address Registers with Shifting

- Syntax: x(An) (x is 16 bits, n is 0-7)
- Example: Move.L 4(A1), D0

Instruction	Before	After
Move.L 4(A1), D0	D0=FFFFFFFF A1= 00001000 $1004=01234567	D0=01234567 A1=00001000 $1004=01234567

Indirect Addressing With Address Registers with Shifting (2)

- Syntax: x(An, Dn.X) (x is 8 bit, n is 0-7)
  x(An, An.X) (X is L or W)
- Example: Move.L 4(A1, A2.L), D0

Instruction	Before	After
Move.L 4(A1, A2.L), D0	D0=FFFFFFFF A1=00001000 A2=00001000 $2004=01234567	D0=01234567 A1=00001000 A2=00001000 $2004=01234567

Absolute Addressing (near)

- Syntax: x (x is 16 bit constant)
- Example: Move.L $1000, D0

Instruction	Before	After
Move.L $1000, D0	D0=FFFFFFFF $1000=01234567	D0=01234567 $1000=01234567

Absolute Addressing (far)

- Syntax: x (x is 32 bit constant)
- Example: Move.L $10000, D0

Instruction	Before	After
Move.L $10000, D0	D0=FFFFFFFF $10000=01234567	D0=01234567 $10000=01234567

Programcounter With Shifting

- Syntax: x(PC) (x is 16 bit constant)
- Example: Move.L $100(PC), D0

Instruction	Before	After
Move.L $100(PC), D0 Assuming PC=$1000	D0=FFFFFFFF $1102=01234567	D0=01234567 $1102=01234567

Programcounter With Index

- Syntax: x(PC, Dn.X) (x is 8 bits, n is 0-7)
  x(PC, An.X) (X is W or L)
- Example: Move.L $100(PC, A1.L), D0

Instruction	Before	After
Move.L $100(PC, A1.L), D0 Assuming PC=$1000	D0=FFFFFFFF $2102=01234567 A1=00001000	D0=01234567 $2102=01234567 A1=00001000

Immediate Addressing

- Syntax: #x (x is 8, 16 or 32 bits)
- Example: Move.L #$10002000, D0

Instruction	Before	After
Move.L #$10002000, D0	D0=01234567	D0=10002000

Addressing with Status Register

- Syntax: SR
  CCR
- It only works with ANDI, EORI and ORI
- If lenght is a byte, the flag register is changed, if it's a word, flag register and the systembyte (supervisor bit should be set)
- Example: ORI #5, CCR
- If CCR was 0000 it becomes 0005

Some words about the stack

- A7 works as a stack pointer
- There are no push/pop instructions
- Use MOVE instead
- To push D0.W : MOVE.W D0, -(A7)
- To pop it back : Move.W (A7)+, D0
- You can use MOVEM to push or pop a whole series of registers
  - MOVEM D0-D4/A0-A2, -(A7)
  - MOVEM (A7)+, D0-D4/A0-A2