The 68000

Lesson 6 – The Execution Block

Some Hints on Structure Programming

- Start writing your program in Pseudo-code
- Structure the main parts of your program
- Go from higher level to lower detail step by step
- Try to relate to what you know from other languages
- Think of REUSABILITY

How to write an If_Else...

- If_1:
  ```
      CMP       D0, D1
      BCC       else_1
  then_1:
      ; code
      BRA       end_if_1
  else_1:
      ; code
  end_if_1:
  ```

How to write a While_Do...

- While_1:
  CMP D0, D1
  BCC end_while_1
  Do_1:
  ;code
  BRA While_1
  end_while_1:

The calling of a function

- When making a function call, a zone is reserved in memory for the function's parameters and local variables.
- Once the function has been executed, we free the memory zone.

EB Structure

- It can be imagined as a 3 part memory block.
- The beginning of it holds the local variables
- The middle holds the control data
- The end holds the parameters
- We access the EB through a Block Pointer, usually A6, that points the beginning of the control data

The parameters zone

- Its size will depend on the number and the type of parameters
- The first parameter will be at the beginning, just next to the control zone

The control zone

- It holds two main data
- First of all a save of the previous block pointer (uses 32 bits)
- Then we have the returning address (32 bits)

The local variables zone

- Size also depends
- For example, if we're representing a
  char s[80]
  you must have at least 80 bytes

A little example

- We are going to write in assembly the following C function, supposing that the EB is already built
- f1 (int x, char *s)

```
{
        int             z=2;
        short  t=3;
        return (*s + x + z + t);
}
```

This is what it happens

- f1:

```
        MOVE.L #2,-4(A6)
        MOVE.W #3,-6(A6)
        MOVE.L 12(A6),A0
        MOVE.B (A0),D0
        EXT.W D0
        EXT.L D0
        MOVE.L 8(A6),D2
        MOVE.L -4(A6),D3
        MOVE.W -6(A6),D4
        EXT.L D4
        ADD.L D2,D0
        ADD.L D3,D0
        ADD.L D4,D0
RTS
```

- This is the EB:
-
```
        Value t (16 bits)
        Value z (32 bits)
A6-> Save of A6 (32 bits)
        Return Address (32 bits)
        Value x (32 bits)
        Address s (32 bits)
```

How to create an EB

- You have to follow several steps:
  - Initialize the parameters zone
  - Then the control zone
  - Finally we prepare enough space for the local variables
- All these operations take place in the stack
- An EB is a block in the stack

Creation example

- f0 ()
  {
  f1 (3,4);
  }

- f0:
  MOVE.L #4, -(A7)
  MOVE.L #3, -(A7)
  BSR f1
  ADD.L #8, A7
  RTS

Creation example cont'd

- f1 (int x, int y)
  {
  int z=23;
  return (x + y + z);
  }

- f1:
  MOVE.L A6, -(A7)
  MOVE.L A7, A6
  MOVE.L #23, -4(A6)
  MOVE.L 8(A6), D1
  MOVE.L 12(A6), D2
  MOVE.L -4(A6), D0
  ADD.L D1, D0
  ADD.L D2, D0
  MOVE.L A6, A7
  MOVE.L (A7)+, A6
  RTS

How to use global variables

- To do this, set the label DATA to start creating variables and CODE when you start writting the program
- You can use
  - DC value (creates a byte, word or longword)
  - DC.B “string” (creates a string of characters)
  - DS number (creates a “number” elements)

Other usefull instructions

- LEA (Load Effective Address)

  LEA <var>, An

- LINK

  LINK An, #data

- UNLK

  UNLK An

What are the Link/unlink instructions?

- LINK An, #value will do:
  SP – 4 -> SP
  An -> (SP)
  SP -> An
  SP + value -> SP
- UNLK An
  An -> SP
  (SP) -> An
  SP + 4 -> SP

An application: using tables

```
char tab[8];

main()
{
        int i;
        for (i=0 ; i<8 ; i++)        {
                tab[i] = i;
        }
        f0(tab);
}
```

```
        DATA
TAB:   DS.B  8
        CODE
MAIN: LINK A6,#-4
        MOVE.L A6, A0
        SUB.L #4,A0
        LEA TAB, A1
        CLR.L      D7
        MOVE.L D7, (A0)
MAIN1: CMP.L #8, D7
        BEQ MAIN2
        MOVE.B 3(A0), (A1)+
        ADD.L #1, D7
        MOVE.L D7, (A0)
        BRA MAIN1
MAIN2:  LEA TAB, A0
        MOVE.L A0, -(A7)
        BSR F0
        ADD.L #4, A7
        UNLK A6
        RTS
```

Using Tables (cont'd)

```
f0 (char *tab)
{
        int i;
        char t[8];
        for (i=0 ; i<8 ; i++)
        {
                t[i] = tab[i];
        }
}
```

```
F0:
        LINK A6, #-12
        MOVE.L A6, A0
        SUB.L #12, A0
        MOVE.L 8(A6), A1
        MOVE.L A6, A2
        SUB.L #8, A2
        CLR.L    D7
        MOVE.L D7, (A0)
F0a: CMP.L #8, D7
        BEQ F0b
        MOVE.B (A1)+, (A2)+
        ADD.L #1, D7
        MOVE.L D7, (A0)
        BRA F0a
F0b: UNLK A6
        RTS
```