APOLLO Core - Fusing Feature

Overview

The APOLLO CPU supports a feature that increases the number of instructions executed per clock. While the core is SuperScalar which means that it can execute two instructions per clock cycle if the destination operand of one instructions isn't the same as the source operand of the subsequent instruction, it can also **bundle two instructions to be executed as a single instruction**. This is possible because the ALU (Arithmetic and Logical Unit) of the APOLLO Core is internally a 3-operand ALU while the 68k only has 2-operand code.

Examples

2-operand code :

`add.l d0,d1` means you add the number in register d0 to the number in `d1` and store the result in `d1`. In C syntax: `d1 = d0 + d1;`.

3-operand code :

`add.l d0,d1,d2` means you add the number in d0 to the number in `d1` and store the result in a third register d2 (or d1 if you wanted to do exactly what `add.l d0,d1` does on a 68k processor). In C syntax: `d2 = d0 + d1;`.

In addition to this the APOLLO ALU has internally more operations than officially supported by the 68k.

We can now exploit these two internal “extras” for instruction bundling and “fuse” two instructions into a single (internal) instruction.

ASM example :

```
move.l d0,d2
add.l  d1,d2
```

In C syntax :

```
d2 = d0 + d1;
```

If you check again what I wrote about 3-operand code, you will see that this is precisely what the single 3-operand instruction add.l d0,d1,d2 does! The APOLLO Core recognises such bundles of 68k instructions and executes them together in a single clock cycle. This is not the same as standard SuperScalar execution because the second 68k instruction depends on the result of the first instruction and thus could not be executed in a single cycle on the 68060.

Since APOLLO is SuperScalar, it can execute these two instructions in addition to yet another instruction or bundle of instructions increasing instructions per clock dramatically.

This also means that in order to optimise code for the APOLLO you would sometimes get even better results by not separating instructions that depend on each other. While on an 68060 you would try to fit an extra independent instruction between the two instructions mentioned above, you should not do so on the APOLLO and just leave it to the core to execute the two instructions together.

Supported fusing combinations

In latest public core (SILVER2).

```
(1)
MOVE.L (An)+,(Am)+
MOVE.L (An)+,(Am)+
=>
MOVE.Q (An)+,(Am)+
```

```
(2)
MOVE.B (d16,An),Dn
EXTB.L Dn
=>
MVS.B  (d16,A0),Dn
```

```
(3)
MOVE.W (d16,An),Dn
EXT.L  Dn
=>
MVS.W  (d16,A0),Dn
```

```
(4)
MOVE.L Dn.Dm
NOT.X  Dm
```

```
(5)
MOVE.L Dn.Dm
NEG.X  Dm
```

```
(6)
MOVE.L Dn.Dm
ADDQ.X #,Dm
```

```
(7)
MOVE.L Dn.Dm
SUBQ.X #,Dm
```

```
(8)
MOVE.L Dn.Dm
ANDI.W #,Dm
```

```
(9)
MOVE.L Dn.Dm
OR.X  Do,Dm
```

```
(10)
MOVE.L Dn.Dm
AND.X  Do,Dm
```

```
(11)
MOVE.L Dn.Dm
ADD.X  Do,Dm
```

```
(12)
MOVE.L Dn.Dm
SUB.X  Do,Dm
```

```
(13)
MOVEQ  #,Dn
OR.X   Dm,Dn
```

```
(14)
MOVEQ  #,Dn
AND.X  Dm,Dn
```

Home | Links | APOLLO |