68000 Address and Bus Error Stack Frame

Version 1.01

Introduction

The bus and address error exceptions in general, and in particular the generated stack frame, are some of the most undocumented and less understood features of the 68000.

When an exception is triggered the processor automatically generates a stack frame that is used to recover from the exception and continue with normal execution. For most exceptions the stack frame contains only the SR (Status Register) and the PC (Processor Counter). Address and bus error are special exceptions classified as Group 0 exceptions that generate and unique and extended stack frame. Hardware reset is also sometimes included as part of the group 0 exceptions, but hardware reset doesn't generate any stack frame at all.

One of the key differences of group 0 exceptions is that they are triggered immediately after the current microcycle is completed. All other exceptions are delayed until the current instruction completes execution. This is one of the reasons that an extended stack frame is generated.

The stack frame, from lower to higher address:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
											R/W	I/N	FUNCTION CODE		
ACCESS ADDRESS HIGH															
ACCESS ADDRESS LOW															
INSTRUCTION REGISTER															
STATUS REGISTER															
PROGRAM COUNTER HIGH															
PROGRAM COUNTER LOW															

R/W (Read Write): Write = 0, Read = 1. I/N (Instruction/Not): Instruction = 1, Not =1

The Access Address is the actual address that was attempted to access at the bus cycle that generated the exception. The value comes from an internal register (AOB) and includes bit 0 that would be set on an Address Error.

We will not cover the full details of the SR in this article. We will just mention that the SR might not be updated by all the changes that could be produced by the current instruction if it would complete without an exception. The behavior is consistent, however. The exact behavior depends on the specific instruction and on which bus cycle the exception was triggered.

The PC field pointing to the middle of the instruction

As described in the official documentation, the PC pushed on the stack frame is advanced by some variable amount from the start of the instruction that was being executed. The reason for this is how the processor's microcode works. During instruction execution, the PC is maintained and updated by a separate register called AU (Arithmetic Unit). As implied by its name, the AU has dedicated hardware to perform additions (and subtractions) that can be used efficiently to increment the PC while the instruction executes. The AU is not used exclusively for the PC. It is also used for computing and updating an operand EA (effective address).

The microcode updates the PC from the AU at different points depending on the exact instruction and microcode sequence. Typically, the microcode increments the PC when fetching extension words. Then the exact current value of the PC depends mainly on how many extension words were already fetched and if the PC was already updated when the exception is triggered.

Bottom word of the stack frame, SSW and undocumented bits

The lowest word of the stack frame is documented as including the SSW (Special Status Word) on the lowermost bits 0 to 5. The content of the rest of the bits on this word, bits 6 to 16, is not officially documented. These undocumented bits are the same bits as the corresponding ones on the value saved on the IR field of the stack frame. The reason, again, is how the microcode works. The microcode pushes the stack frame in a non consecutive order. Furthermore, some registers are not written directly to external memory but instead they are transferred through an intermediate register called FTU (Field Translation Unit). The microcode first transfers the current IR, or more precisely, the IRD (Instruction Register Delay) to the FTU. Later it transfers the SSW that is 5 bits wide. Then the remaining bits of the FTU retain the previous value from the IRD.

The SSW includes three fields. Two well-known fields are the function code and the R/W signals when the bus or address error occurred. The third one is named I/N and is set if the processor is processing a group 1 or a group 0 exception. In practice, the stack frame would not be generated if the processor were already on a group 0 exception; because a group 0 exception inside another one is considered a double fault and will halt the processor.

IR field might have the next instruction OPCODE

The IR field will normally be the opcode of the instruction that executed the bus cycle that caused the exception. However, on some cases it could be the opcode of the next instruction. Technically, the content of this field is a copy the IRD when the group 0 exception microcode starts executing.

As we elaborated on an earlier article, the 68000 has three registers used for the instruction opcode. The IRC (Instruction Register Cache) is the storage for the last word pre fetched. The IR holds the instruction opcode being decoded. And the IRD holds the opcode that is being currently executed. By the end of the current instruction, the IR is forwarded to the IRD. The exact prefetch flow and the

distinction between IR and IRD are not very important in this context. See the mentioned article about more details of the instruction prefetch. For our purposes, however it is very relevant exactly when this IRD forwarding is performed.

The IRD is normally updated at the last microcycle of the instruction. If the bus or address error happens on this last microcycle, the IRD forwarding is still performed because the exception doesn't interrupt the current microcycle. Then once the exception microcode takes control, the IRD was already updated to the opcode of the next instruction that would have been executed if there weren't an exception. As mentioned, it is the content of this IRD register that later is copied to the IR field of the stack frame.

The I/N field might be set

It has been further observed that on some circumstances, not only the IR field contains the opcode of the next instruction, but also the I/N field is set indicating, or seeming to indicate, that the processor wasn't executing an instruction when the exception was triggered. Normally the I/N field would be set only when the processor was already processing a group 1 exception, such as when a bus or address error occurs when saving the PC or the SR during an interrupt. It is not expected to be set when processing an instruction. The reason that it would be set in some circumstances, is once more again, how the microcode works.

As explained in the previous section, the processor updates its state for the next instruction before it actually completes. This includes updating the TVN (Trap Vector Number) latch. The TVN is used slightly differently depending on the exception group. For group 1 exceptions, besides specifying the vector number, it is also the flag to process an exception after the current instruction completes.

Group 0 exceptions, on the other hand, are processed after the current microinstruction and do not update or modify the TVN latch until later, once the exception is already processing.

Consider the following code sequence (assembled machine code is included because it would be useful for the waveform below):

```
7001  moveq    #1,d0
2040  move.l   d0,a0
3100  move.w   d0,-(a0)
4AFC  illegal
```

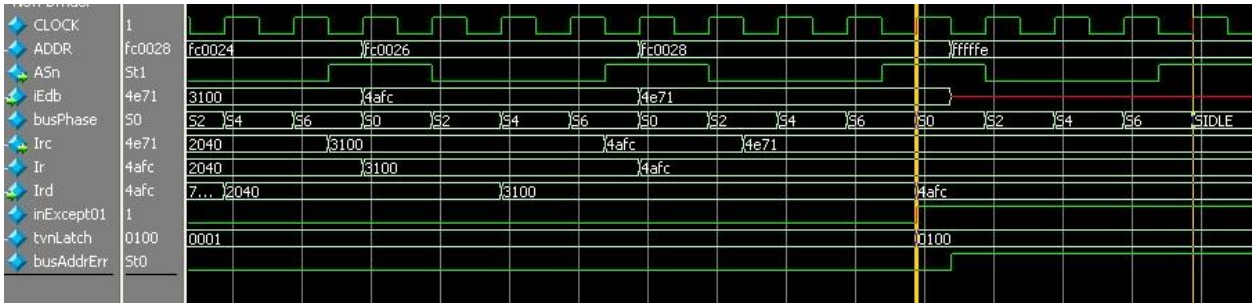
When the write to memory occurs as part of the third instruction, the EA is odd which will provoke an address error. In this particular instruction, using pre decrement addressing mode, the write bus cycle is performed at the end of the instruction. Then, the TVN latch is updated when the write bus cycle starts, before the group 0 exception is even detected. In addition, because in this particular code sequence the next instruction is illegal, the TVN is set for an illegal exception.

The address error overrides the illegal exception because it has higher priority but it is not processed until the bus cycle is aborted at the end of the current microinstruction. Note that the TVN is unaffected and remains indicating an illegal exception, but the illegal exception is actually never processed. Lastly, once the exception microcode starts, the I/N field in the SSW is set because the TVN latch indicates a group 1 exception. Note that this would not happen if the following instruction would be a regular one, not unless another group 1 exception, like Trace or Interrupt was being triggered.

It might seem surprising that both the IRD and the TVN are updated before the current instruction actually completes. From a software processing point of view, it could be expected that first the instruction completes, only then the state for the next instruction is updated, and finally the next instruction or exception is processed. But hardware doesn't work like that. There is certainly no gap between instructions, and there isn't an exact edge that absolutely separates one instruction from the other. For pipelining reasons and because of internal delays, some updates must be performed some cycles ahead. In this case specifically, the processor cannot update the TVN latch at the very last cycle. The TVN latch determines the address of the next microcode and the processor needs a couple of cycles to decode the address and the internal microcode ROM output.

FX68K simulation waveform

This is a simulation waveform of my FX68K FPGA core performing the above code. The waveform for a real 68000 would be slightly different, mainly because synchronization is different. But the result would be exactly the same.



The yellow vertical marks enclose the faulty bus cycle that attempts to access an odd address. Note the flow of the opcodes through the prefetch. Both the IRD and the TVN latch are updated at the start of the last microcycle of the MOVE instruction, coinciding also with the start of the bus cycle. The address error is signaled internally rather early; this is needed to abort the bus cycle at the external signals. However, internally the bus cycle continues, the MOVE instruction is still executing and the actual exception microcode has not yet started. Exception processing, including updating the SSW as one of its first tasks, starts a couple of cycles later (not shown on the waveform). By then, IRD already has the ILLEGAL instruction's opcode, and the TVN latch indicates a group 1 exception.