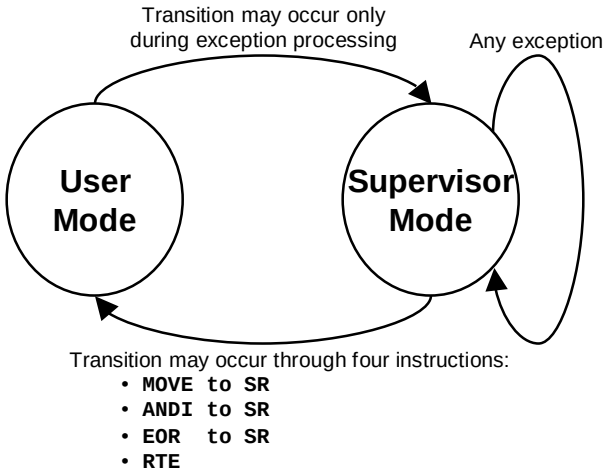
Lecture 9: Exception processing

- **Privilege states and exceptions**
- **Exception taxonomy**
- **Exception processing in detail**
- **Hardware-initiated exceptions**
- **Interrupts**
  - Vectored interrupts
  - Auto-vectored interrupts
- **Software-initiated exceptions**

Privilege states

- **The MC68000 provides two states of privilege**
  - Supervisor (S-bit in SR is 1)
  - User (S-bit in SR is 0)
- **Each state has its own stack pointer and stack**
  - User Stack Pointer
  - Supervisor Stack Pointer
- **A few instructions are only available in supervisor mode**

```
- AND  #data,SR      - MOVE An,SSP
- EOR  #data,SR      - RESET
- OR   #data,SR      - RTE
- MOVE <ea>, SR      - STOP
- MOVE SSP,  An
```

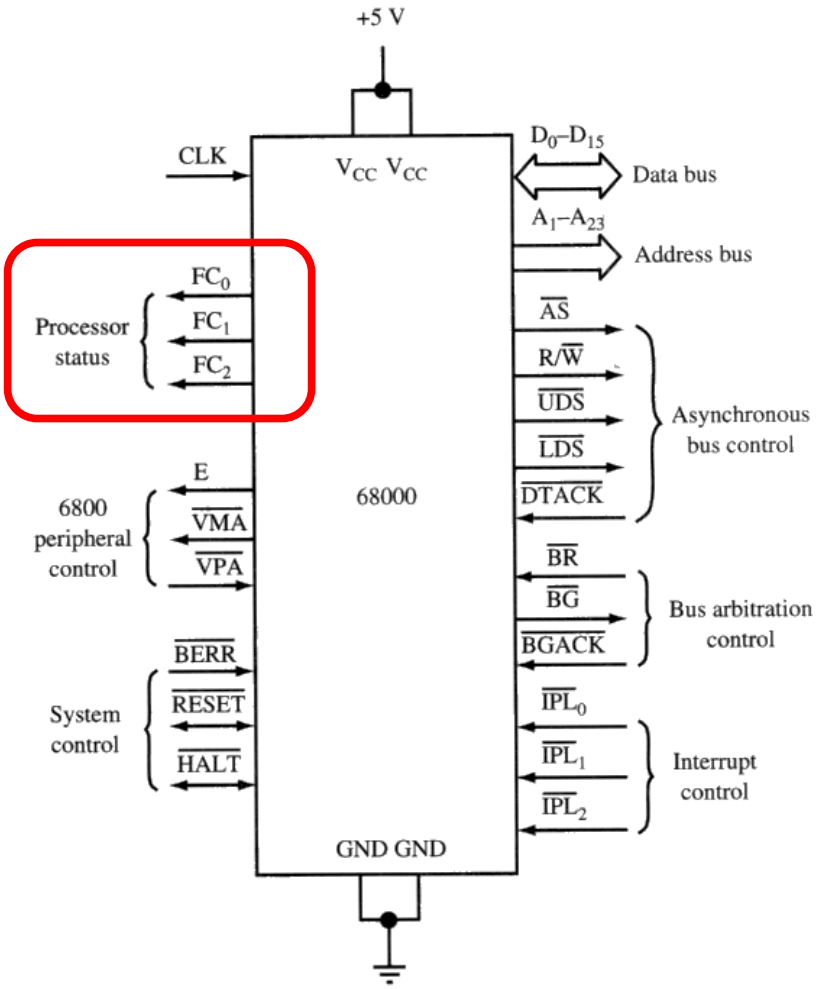


Function code output pins

- **The state of the CPU is indicated to external circuitry by three *function code output pins*: FC0, FC1, FC2**

FC2	FC1	FC0	Memory access type
0	0	0	Undefined --reserved
0	0	1	User data
0	1	0	User Program
0	1	1	Undefined --reserved
1	0	0	Undefined --reserved
1	0	1	**Supervisor data***
1	1	0	**Supervisor program****
1	1	1	1ACK space (CPU space)

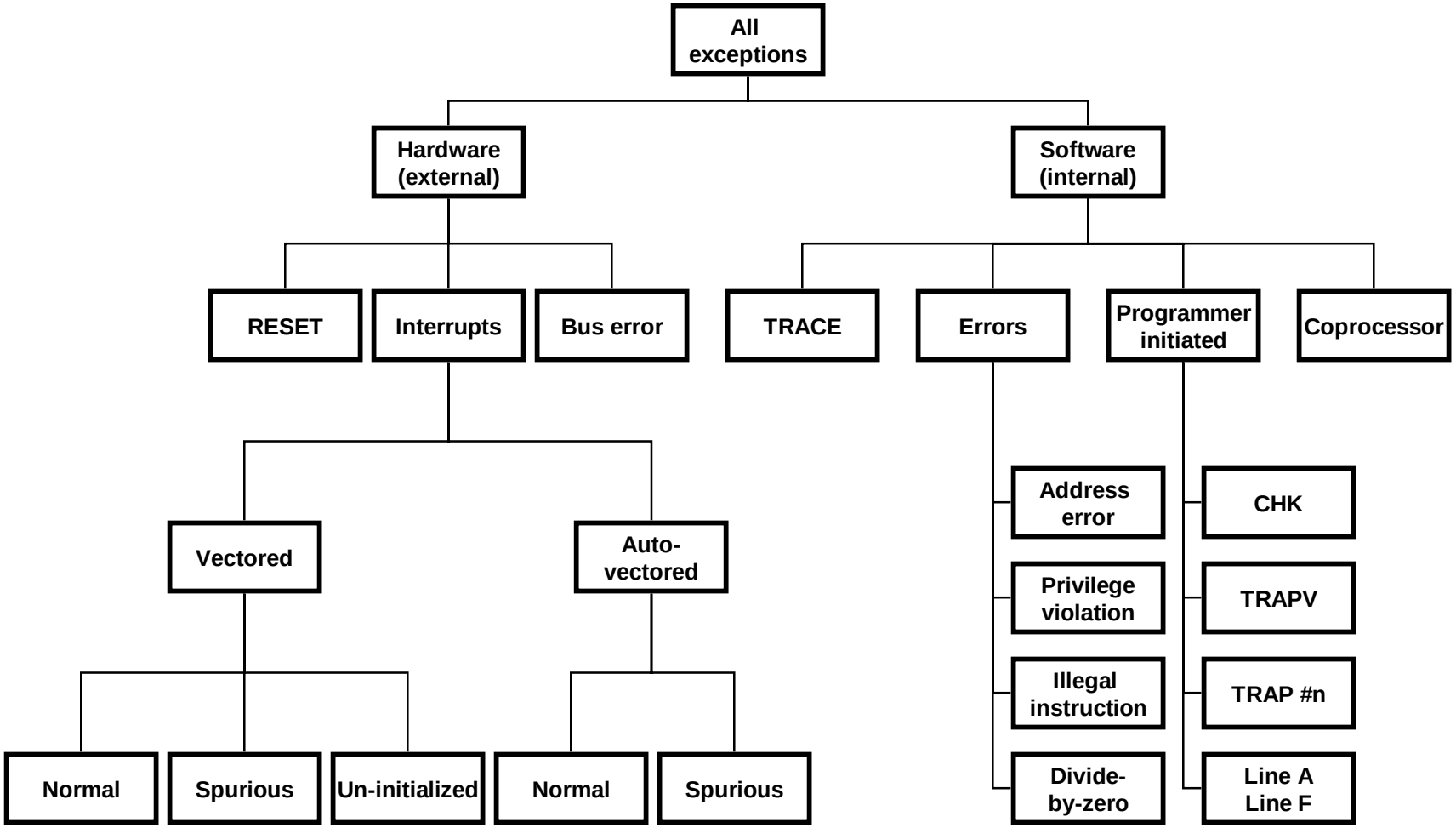
* System is in supervisor mode and accessing data from memory
** System is in supervisor mode and accessing an instruction from memory



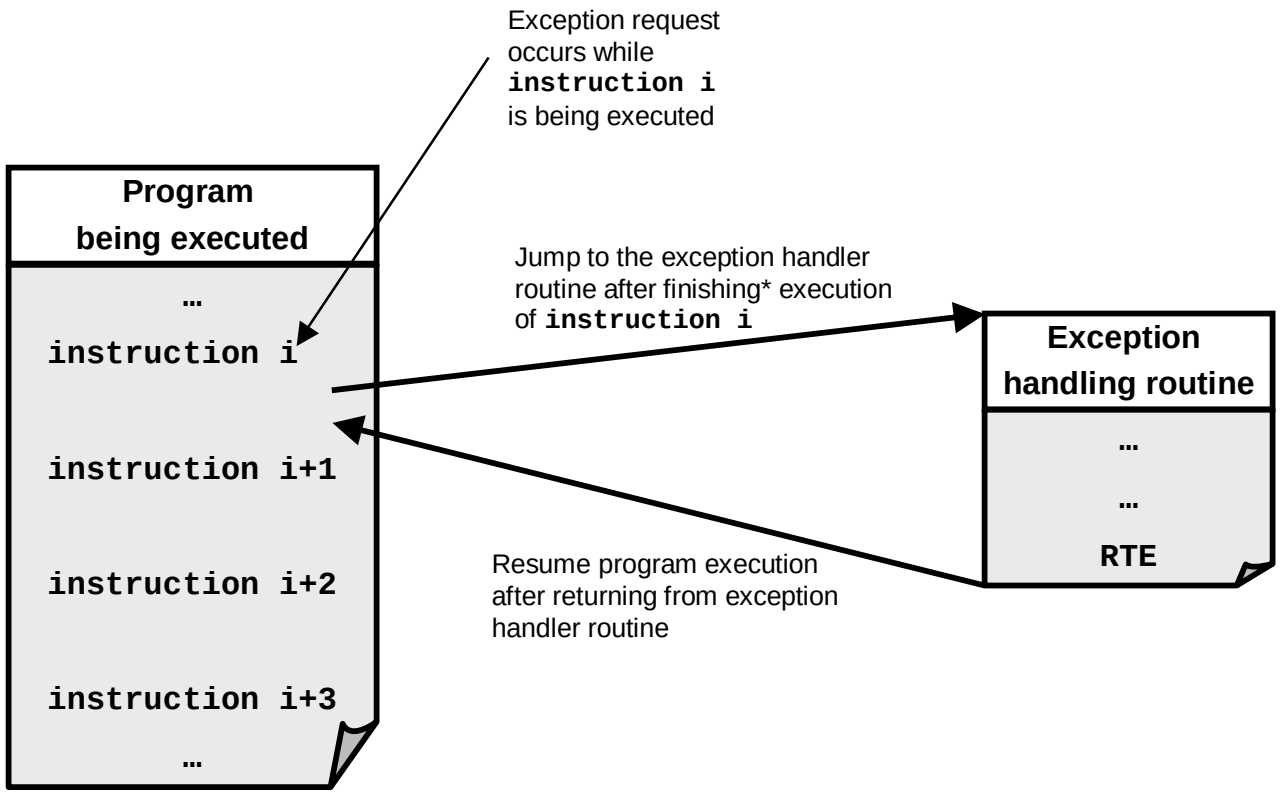
Exceptions

- **Exceptions can be viewed as *operating system calls***
- **Exceptions can be made by either**
  - the programmer requesting an OS service (I/O for instance) or
  - the 68000 in response to certain kinds of SW or HW errors
- **Each type of exception as its own *exception handler* to deal with the condition that triggered the exception**
- **Exception handlers are *very similar* to subroutines BUT**
  - Exceptions don't require an explicit address (it is determined by a vector #)
  - Exceptions save PC and SR (and more) to the Supervisor Stack
  - Exceptions set the S-bit to 1, and subroutines do not alter the SR
  - A RTE is used to return instead from the usual RTS
  - Nesting of exceptions is prioritized
  - Exceptions are typically written by the *systems programmer*
- **Exceptions can be external or internal**

Exception taxonomy



Exception processing



*With the exception of RESET, BUS ERROR or ADDRESS ERROR exceptions, which initiate exception processing within two clock cycles

Exception processing in more detail

- **When the 68000 receives an exception, the following procedure is performed**
  - Save PC and SR to the Supervisor Stack
  - Determine the address of the exception handler
  - Execute the exception
  - Restore PC and SR from the Supervisor Stack

- **To determine the address of the exception handler, the 68000 uses a VECTOR TABLE**
  - The vector table is stored at $00 0000 to $00 03FF
  - Each element in the vector table consists of 4 bytes that specify the address of the exception handler
  - Each exception is associated with an index of the vector table called a vector number
  - The handler address is stored in a memory location pointed by the vector number × 4

Jump to an exception handler

```
[temp_register]   ← [SR]
[S]               ← [1]
[T]               ← [0]
get_vector_number
exception_vector  ← vector_number×4
handler_address   ← [M(exception_vector)]
[SSP]             ← [SSP]-4
[M(SSP)]          ← [PC]
[SSP]             ← [SSP]-2
[M(SSP)]          ← [temp_register]
[PC]              ← handler_address
```

Return from an exception handler

```
[SR]              ← [M(SSP)]
[SSP]             ← [SSP]+2
[PC]              ← [M(SSP)]
[SSP]             ← [SSP]+4
```

Vector number	Memory address	Memory contents
0	$0000	Reset, initial supervisor stack pointer
1	$0004	Reset, initial program counter value
		...
N	(N)×4	Starting address of the N^{th} handler
N+1	(N+1)×4	Starting address of the $(N+1)^{th}$ handler
N+2	(N+2)×4	Starting address of the $(N+2)^{th}$ handler

The vector table

vector number (Decimal)	address (Hex)	assignment
0	0000	RESET: initial supervisor stack pointer (SSP)
1	0004	RESET: initial program counter (PC)
2	0008	bus error
3	000C	address error
4	0010	illegal instruction
5	0014	zero divide
6	0018	CHK instruction
7	001C	TRAPV instruction
8	0020	priviledge violation
9	0024	trace
10	0028	1010 instruction trap
11	002C	1111 instruction trap
12*	0030	not assigned, reserved by Motorola
13*	0034	not assigned, reserved by Motorola
14*	0038	not assigned, reserved by Motorola
15	003C	uninitialized interrupt vector
16-23*	0040-005F	not assigned, reserved by Motorola
24	0060	spurious interrupt
25	0064	Level 1 interrupt autovector
26	0068	Level 2 interrupt autovector
27	006C	Level 3 interrupt autovector
28	0070	Level 4 interrupt autovector
29	0074	Level 5 interrupt autovector
30	0078	Level 6 interrupt autovector
31	007C	Level 7 interrupt autovector
32-47	0080-00BF	TRAP instruction vectors**
48-63	00C0-00FF	not assigned, reserved by Motorola
64-255	0100-03FF	user interrupt vectors

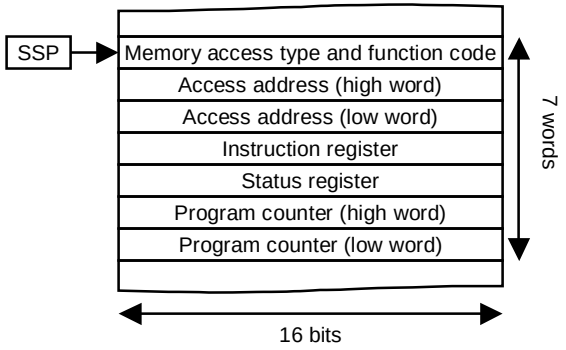
NOTES:
* No peripheral devices should be assigned these numbers
** TRAP #N uses vector number 32+N

Exception types and priorities

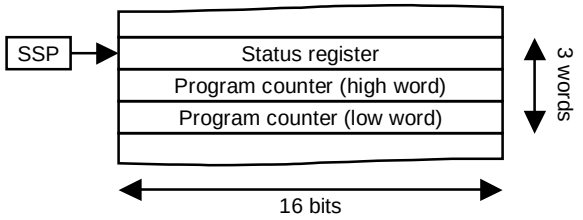
- **The exceptions in the 68000 are divided into three groups, according to their priority and characteristics**
  - Group 0 exceptions processed before Groups 1 and 2
  - Group 1 exceptions processed before Group 2
- **GROUP 0 exceptions**
  - Basically mean that something has gone seriously wrong with the system
  - For this reason, more detailed information is saved on the stack to assist *diagnosis*
  - You cannot return from a Group 0 exception with the RTE command since the information stored on the stack is different than what RTE expects!
- **GROUP 1 exceptions**
  - generated by traces, interrupts, illegal op-codes or privilege violations
- **GROUP 2 exceptions**
  - Occur as part of the normal instruction-execution sequence of a program

Group	Exception	Time at which processing begins
0	Reset Bus error Address error	*Exception processing begins within two clock cycles*
1	Trace Interrupt Illegal op-code Privilege	*Exception processing begins before the next instruction*
2	TRAP TRAPV CHK Divide-by-zero	*Exception processing is started by normal instruction execution*

Information saved on stack for Group 0



Information saved on stack for Groups 1,2

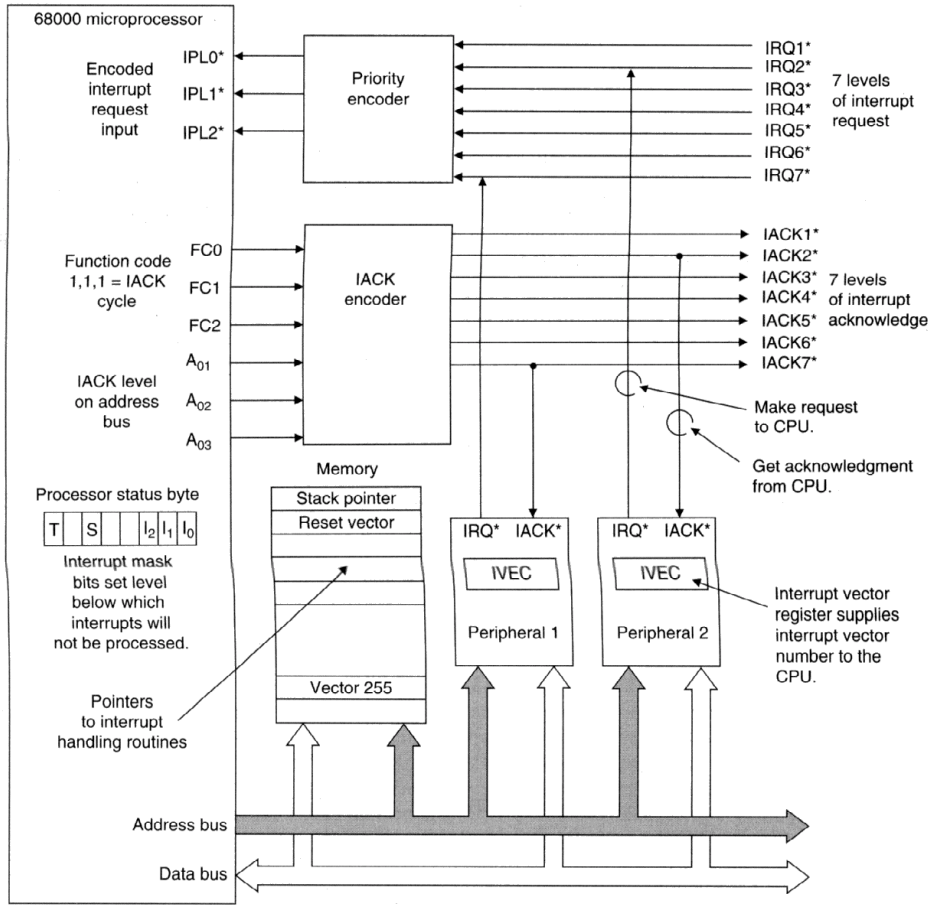


Hardware-initiated interrupts

- **RESET**
  - Takes place only under two circumstances
    - Power-up
    - Total and irrecoverable system collapse
  - RESET has the highest priority and will be processed before ANY other exception that is pending or being processed
  - Detected by a low RESET* pin on the 68000
  - On a RESET exception, the 68000 performs the following operations
    - `[SR]` ← `$2700` (S=1, T=0 and interrupt mask level set to 7)
    - `[SSP]` ← `[$0000]` (initialize Supervisor Stack with the first element in the vector table)
    - `[PC]` ← `[$0004]` (Initialize PC with the second element in the vector table)
    - Resume execution at the position pointed by PC
- **BUS ERROR**
  - Raised by failure of the system to complete a bus cycle
    - Illegal memory access: the processor tried to access an address not populated by memory
    - Fault to memory access: if error-detecting memory is used
    - Failure to assert VPA*: used for auto-vectored interrupts (details later)
    - Memory privilege violation: when the 68000 uses some form of memory management
  - Detected by a low BERR* pin on the 68000
- **INTERRUPTS… next slides**

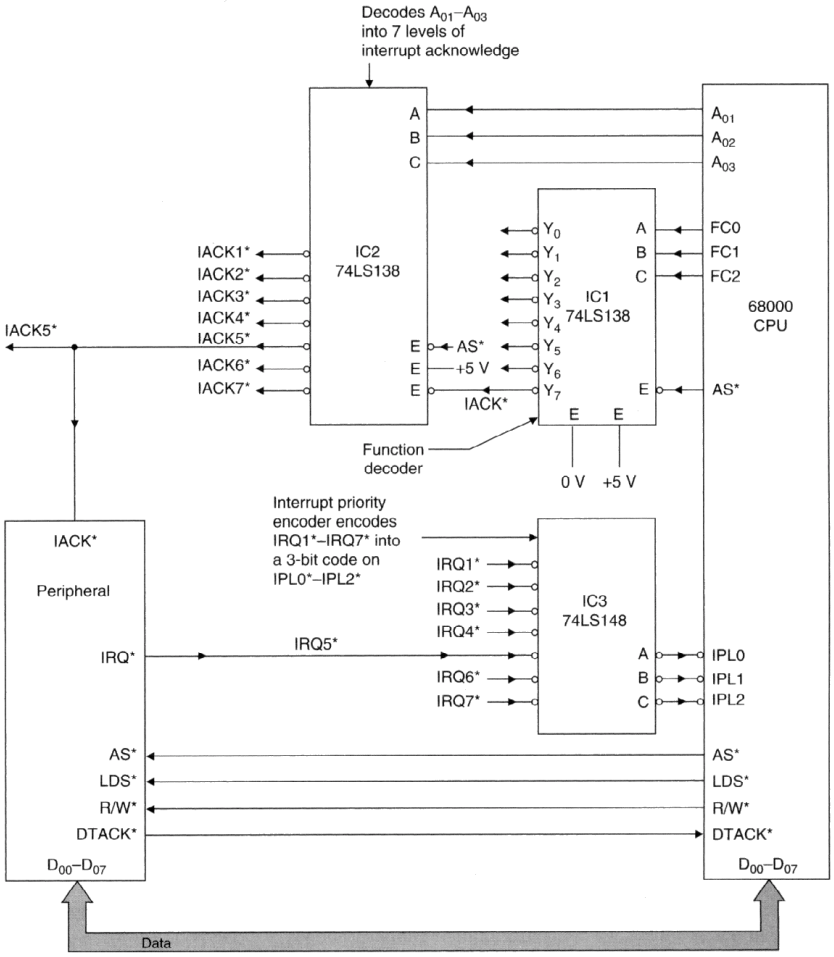
Interrupts

- **The 68000 provides two interrupt schemes**
  - **Vectored**: intended for modern 16-bit peripherals
  - **Auto-vectored**: intended for older 8-bit peripherals
- **There are seven levels of interrupts available**
- **The sequence of operations during an interrupt request is the following**
  - A peripheral requires attention by asserting its interrupt request output (IRQ*)
  - The priority encoder produces a 3-bit code with the highest IRQ* line active and passes it to the 68000 on the IPL0*-IPL3* inputs
  - The 68000 compares the level of the interrupt with the interrupt mask flag ($I_2I_1I_0$) in the SR.
  - If the requested input is greater than ($I_2I_1I_0$), the interrupt is serviced, otherwise it is ignored
  - If the 698000 decides to service the interrupt:
    - The code 111 is placed on ($FC_2FC_1FC_0$) to inform the system that an interrupt is about to be serviced
    - The priority of the interrupt is placed on ($A_3A_2A_1$)
    - ($FC_2FC_1FC_0$) and ($A_3A_2A_1$) are passed to an interrupt acknowledge decoder which asserts one of the seven IACK* lines
  - The asserted IACK* line informs the interrupting device that it is about to be serviced
  - The remaining steps are dependent on the type of interrupting device (vectored or auto-vectored) and are detailed in the next two slides



Vectored interrupts

- **Vectored interrupts are intended for peripherals that can provide an 8-bit vector number (256 values) to the 68000**
  - This vector number is stored in register in the peripheral (IVEC)
  - **It is the programmer's responsibility** to initialize the device with the appropriate vector number!
- **After the appropriate IACK* line is asserted by the 68000, the following operations are performed**
  - The peripheral whose interrupt level matches the asserted IACK* will "know" that it is going to be serviced
  - The peripheral then writes the IVEC vector onto the data bus (D_7D_0) and asserts the DTACK* line (DTACK stands for Data Transfer Acknowledge)
  - the active DTACK* terminates the IACK cycle and the 68000 will execute the interrupt handler pointed by the vector fetched from (D_7D_0)
- **There are two variations to this procedure**
  - If DTACK* is not asserted, BERR* (Bus Error) must be asserted by the external hardware to force a *spurious interrupt exception*
  - If the peripheral has not been initialized with an appropriate vector, it should place $0F on the data bus to force an *uninitialized interrupt vector exception*

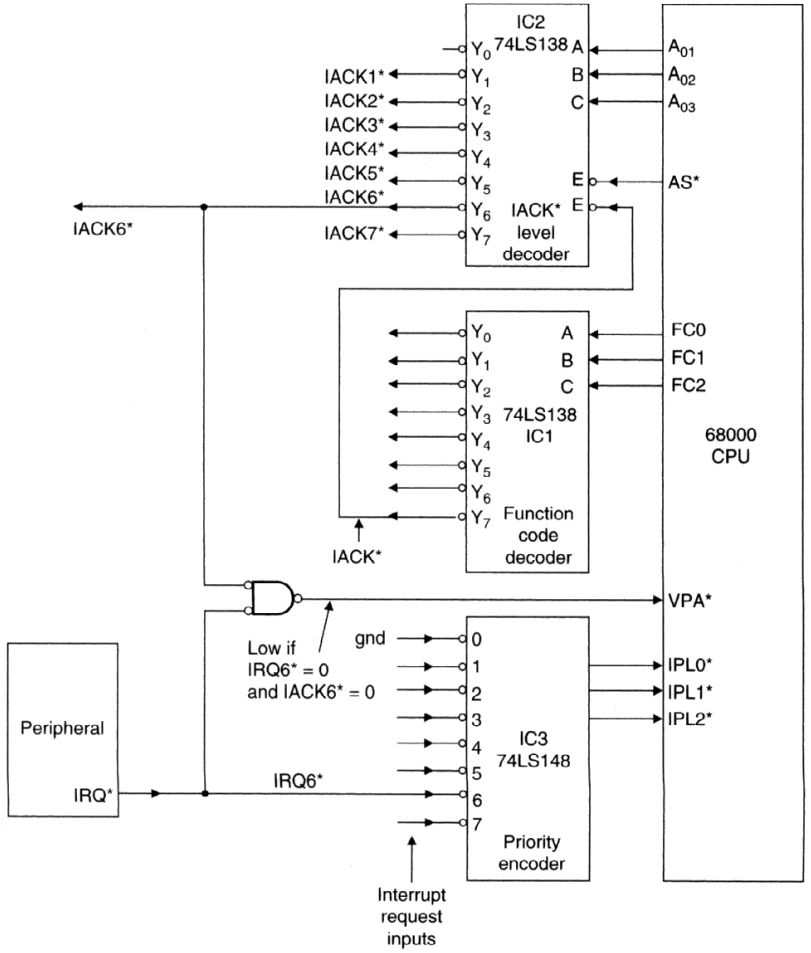


Auto-vectored interrupts

- **Auto-vectored interrupts are intended for earlier peripherals designed for 8-bit processors that cannot provide a vector during an IACK cycle**
- **After the appropriate IACK* line is asserted by the 68000, the following operations are performed**
  - The interrupting device will assert the VPA* line (Valid Peripheral Address)
  - Upon receiving an asserted VPA* line, the 68000 assumes the peripheral is a 6800-series and then
    - ignores the contents of (D_0-D_7)
    - internally generates an interrupt vector from the priority level of the IRQ* line that was asserted
    - the 68000 reserves vector numbers $19-$1F for auto-vectored interrupts on IRQ1*-IRQ7*:

25	0064	Level 1 interrupt autovector
26	0068	Level 2 interrupt autovector
27	006C	Level 3 interrupt autovector
28	0070	Level 4 interrupt autovector
29	0074	Level 5 interrupt autovector
30	0078	Level 6 interrupt autovector
31	007C	Level 7 interrupt autovector

  - When several peripherals are assigned to the same IRQ* level, then the 68000 cannot distinguish between then and the appropriate auto-vectored handler routine MUST poll each of the possible requesters and read their interrupt status registers



Software-initiated interrupts

- **TRACE**
  - Forced when the T-bit of the SR is set to 1
  - When T=1 a trace exception is generated after each execution of an instruction
  - This enables the programmer to step through the execution of a program
- **Address error:**
  - Occurs when the 68000 attempts to access a 16- or 32-bit longword at an odd address
- **Privilege violation**
  - Occurs when the processor is in user mode and attempts to execute a instruction reserved for the supervisor state
- **Illegal instruction**
  - Occurs when the CPU fetches an op-code from memory that corresponds to an unimplemented instruction
    - This typically occurs when the effective address of a branch instruction is computed wrong
- **Divide-by-zero**
  - Occurs when a number is divided by zero

Software-initiated interrupts

- **CHK**
  - Used to check against bounds
    - For example, checking of array indexes against the boundaries of the array
  - Generated by the assembly instruction **`CHK <ea>,Dn`**
  - For example, the instruction **`CHK D1,D0`** will generate a CHK exception if
    - **`[D0(0:15)]<0`** or
    - **`[D0(0:15)]>[D1(0:15)]`**
- **TRAPV**
  - Generated by the assembly instruction TRAPV, which forces the exception if the V-bit of the CCR is set to 1
- **Emulators**
  - Op-codes whose four MSBs (bits 12 to 15) are %1010 ($A) or %1111 ($F) are unimplemented in the 68000, but they are not treated as illegal instructions!
  - Line A instructions
    - Used to emulate instructions missing in the 68000
  - Line F instructions
    - Similar to Line A, but is associated with a different vector number
    - Typically used in 68020 to emulate a co-processor when one is not present in hardware

Software-initiated interrupts: TRAPS

- **TRAPS are the most useful software user-initiated exceptions available**
- **TRAPS are not different from line A or line F exceptions except for the vector number associated with them**
- **Generated with the instruction `TRAP #0` to `TRAP #15`, which are associated with vector numbers 32 to 47 decimal**
- **TRAP is normally used to provide portable code between 68000-based systems with different peripherals**
- **At first it may seem that the 68000 is limited to 16 TRAP operations**
  - This is not the case since it is possible to pass a integer to the trap handler in a data register
  - Within the trap handler, the integer is used as an index into a jump table that points to the desired routine

Exception processing flowchart