

# CSCE 313

## Embedded Systems Design

---

2005/4/14

### Lecture Notes – Week 11

### Exceptions, Interrupts, Privilege States

© 2004 James P. Davis. Ph.D.

Many Slides © 1992 Prentice-Hall (Robert O. Pettus, Ph.D. & Joe Byrd, Ph.D.)

---

## Outline

---

- Where we are (1<sup>st</sup> half of play):
  - ✓ We have made a complete high-level pass through the 68000 architecture and programming model: (1) studied instructions, and basic program construction, (2) studied architecture for instruction fetch, decode, execute operations, (3) written and debugged programs using WISM68 simulator/emulator on PC.
  - ✓ We understand that issues of program efficiency (in terms of trading off execution speed versus systems resource usage) is an important aspect of good engineering practice, in addition to following principles of good programming methodology and software engineering practice.
- Where we are going (2<sup>nd</sup> half of play):
  - ✓ We now make a 2<sup>nd</sup> iteration through the subject matter, going deeper into the concepts, refining our model and understanding of the architecture and programming model.
  - ✓ We'll explore more advanced feature/functions of the process architecture, and the programming model to use these capabilities to write effective programs: (1) *interrupts*, (2) *exceptions* and *traps*, (3) *user/supervisor mode* state changes.
  - ✓ We'll focus on exploring 3 different uses of the 68K processor for building systems: (1) putting an *operating system* on it, (2) *interfacing* the 68K with available lab hardware, (3) talking about the system requirements for *reactive real-time processing* control applications.

# Introduction to Exceptions

- Sequential programs specify the sequence but not a specific time of execution; real-time applications require time synchronization.
- Program should be able to respond to external events; process control applications require this.
- A mechanism is needed to respond to abnormal conditions that exist because of hardware failures or program errors.

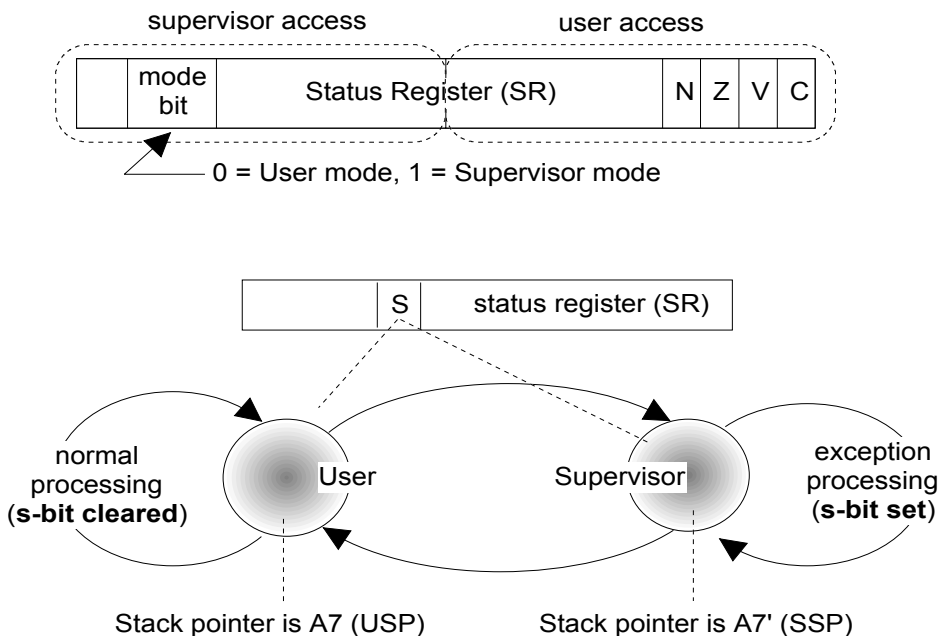
**THE EXCEPTION PROCESS IS THE ANSWER!**



© 2002 Dr. James P. Davis Page 3

## Mode Selection and Control

Source Pettus et al. © 1993 Prentice-Hall Publishers



© 2002 Dr. James P. Davis Page 4

# Privileged Instructions

Source Pettus et al. © 1993 Prentice-Hall Publishers

- Instructions that can only be executed in the Supervisor Mode.
- These instructions allow for operations in the Status Register (all 16-bits).
- Special instructions that generate exceptions (TRAPV is a good example)

INSTRUCTION	OPERATION
ANDI #<immed>, SR	Logical AND immediate word to SR
EORI #<immed>, SR	Logical Ex-OR immediate word to SR
ORI #<immed>, SR	Logical Inc-OR immediate word to SR
MOVE <ea>, SR	Move word from effective address to SR
MOVE An, USP	Move long address from An to user system stack pointer
MOVE USP, An	Move value from user system stack pointer to An
RESET	Asserts bus reset line
RTE	Return from exception process
STOP	Stops processor execution

Table 9-1. Privileged instructions for the 68000

RETURN INSTRUCTION	RETURN OPERATION
RTE	Return from exception process service routine. Clears S-bit in SR.
ANDI.W #0DFFF, SR	Clear S-bit in SR. All other bits unchanged
EORI.W #02000, SR	Clears S-bit in SR (always set when instruction is executed)

Table 9-2. Returns to user mode



© 2002 Dr. James P. Davis Page 5

# Exception Vectors

Source Pettus et al. © 1993 Prentice-Hall Publishers

- Addresses \$000000-\$0003FF in the address space are reserved for exception vectors.
- The vector address contains the address of the exception routine; therefore, the “vector” points the system to the location of the action to be executed.
- Vector Number x 4 = Vector Address (because each address is 4 bytes long)

VECTOR NUMBER	ADDRESS (HEX)	ASSIGNMENT
0	000	RESET (REST ART)
	004	RESET (REST ART)
2	008	BUS ERROR
3	00C	ADDRESS ERROR
4	010	ILLEGAL INSTRUCTION
5	014	ZERO DIVIDE
6	018	CHK INSTRUCTION
7	01C	TRAPV INSTRUCTION
8	020	PRIVILEGE VIOLATION
9	024	TRACE
10	028	LINE 1010 EMULATOR
11	02C	LINE 1111 EMULATOR
12	030	UNASSIGNED, RESERVED
13	034	UNASSIGNED, RESERVED
14	038	UNASSIGNED, RESERVED
15	03C	UNINITIALIZED INTERRUPT
16-23	040-05F	UNASSIGNED, RESERVED
24	060	SPURIOUS INTERRUPT
25	064	LEV1 INTERRUPT AUTOVECTOR
26	068	LEV2 INTERRUPT AUTOVECTOR
27	06C	LEV3 INTERRUPT AUTOVECTOR
28	070	LEV4 INTERRUPT AUTOVECTOR
29	074	LEV5 INTERRUPT AUTOVECTOR
30	078	LEV6 INTERRUPT AUTOVECTOR
31	07C	LEV7 INTERRUPT AUTOVECTOR
32-47	080-0BF	TRAP INSTRUCTIONS
48-63	0C0-0FF	UNASSIGNED, RESERVED
64-255	100-3FF	USER INTERRUPTS

Table 9-3. Vector assignments



© 2002 Dr. James P. Davis Page 6

# Exception Handling Routines

Source Pettus et al. © 1993 Prentice-Hall Publishers

- Initiated by hardware or software.
- The current PC (program counter) along with other information is pushed to the system stack at the time of an exception.
- RTE is a special return instruction for exceptions. Takes care of restoring PC, status information, etc.
- Exception routine (or exception process) runs in the Supervisor Mode. (whereas subroutines may run in either mode.)

INSTRUCTION	CONDITION	OPERATION
TRAP #n	none	Trap sequence using Trap Vector #0 - #15
TRAPV	V = 1	Trap sequence for overflow at vector TRAPV
CHK <ea>, Dn	Exceeded check range	Trap sequence using CHK vector when value in Dn greater than <ea> or less than 0
DIVS <ea>, Dn DIVU <ea>, Dn	(ea)=0	Trap sequence using Zero Divide vector when divisor is 0
RTE	End of service	Return from exception routine to the program in which exception occurred

Table 9-5. Exception instructions

PROCESS STEP	OPERATIONS
1	Saves the current SR. Sets the S-bit (Bit 13) and clears T-bit (Bit 15) of SR. Puts processor in supervisor mode and inhibits tracing.
2	Pushes current PC onto supervisor stack. Saves return address.
3	Pushes saved SR (saved in Step 1) on supervisor stack.
4	Gets vector number; loads vector address into PC. Enters service routine for the exception.

Table 9-6. Exception processing steps



© 2002 Dr. James P. Davis Page 7

## Exception and Interrupt Vector Types

Source Pettus et al. © 1993 Prentice-Hall Publishers

- External (H/W generated)
  - ✓ Reset
  - ✓ Bus Error
  - ✓ Interrupts
    - ✦ Autovector
    - ✦ Vector-supplied
- Internal (S/W generated)
  - ✓ Execution error
  - ✓ Illegal Instruction
  - ✓ Instructions (TRAP, TRAPV, etc)
  - ✓ Trace

PRIORITY GROUP	EXCEPTION TYPE	PROCESSING
0 (Highest)	Reset Bus Error Address Error	Exception processing begins within two clock cycles of the condition
1	Trace Interrupt Illegal Op-Code Privilege	Exception processing begins at the end of the current instruction
2 (Lowest)	TRAP TRAPV CHK Divide by Zero	Exception processing begins at normal execution of instruction

Table 9-7. Exception priorities

An *autovectored* mode, in simple applications, lets the processor assign the vector for the service routine.

The more general *vectored* mode requires that the requesting device provide information for its vector assignment.

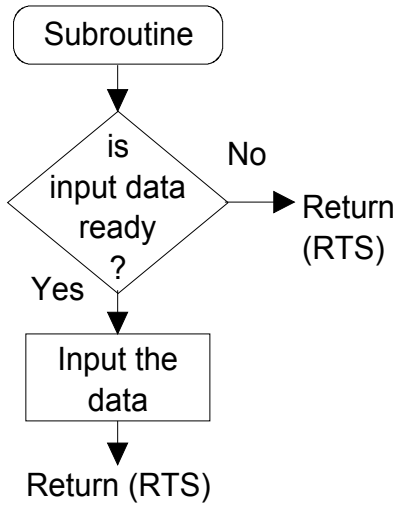
Seven levels of priority are provided with the highest level being “unmaskable”, i.e., it cannot be masked by the Interrupt Mask.



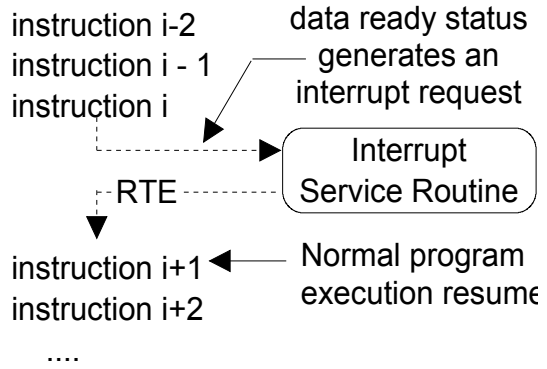
© 2002 Dr. James P. Davis Page 8

# Interrupts versus Polling

Source Pettus et al. © 1993 Prentice-Hall Publishers



(a) a polling process



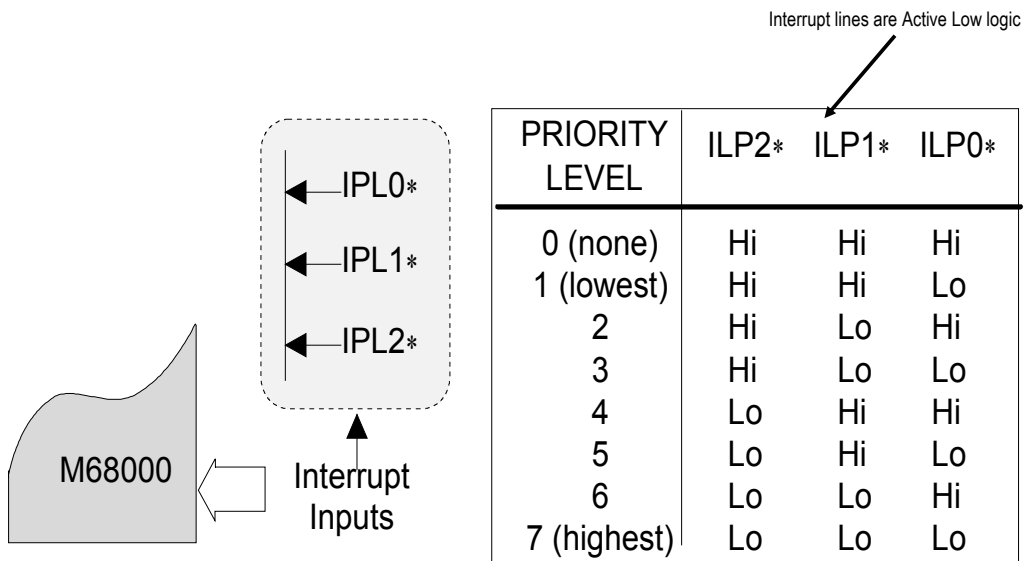
(b) an interrupt-driven process



© 2002 Dr. James P. Davis Page 9

# Interrupt Bus Signals

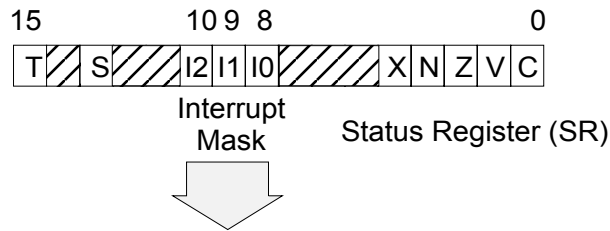
Source Pettus et al. © 1993 Prentice-Hall Publishers



© 2002 Dr. James P. Davis Page 10

# The Interrupt Mask

Source Pettus et al. © 1993 Prentice-Hall Publishers



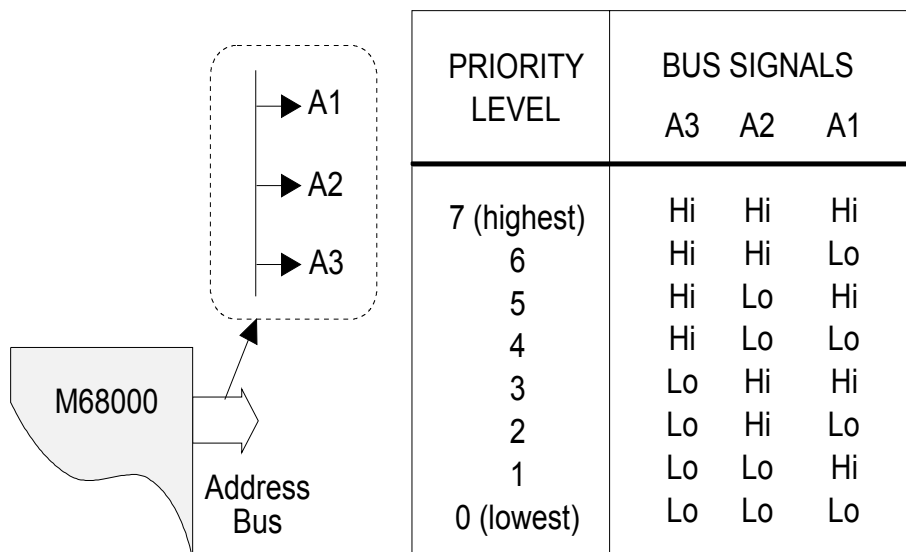
PRIORITY LEVEL	INTERRUPT MASK			RECOGNIZED LEVELS
	I2	I1	I0	
7 (highest)	1	1	1	7
6	1	1	0	7
5	1	0	1	6, 7
4	1	0	0	5, 6, 7
3	0	1	1	4, 5, 6, 7
2	0	1	0	3, 4, 5, 6, 7
1	0	0	1	2, 3, 4, 5, 6, 7
0 (lowest)	0	0	0	1, 2, 3, 4, 5, 6, 7



© 2002 Dr. James P. Davis Page 11

# Interrupt Priority on Bus

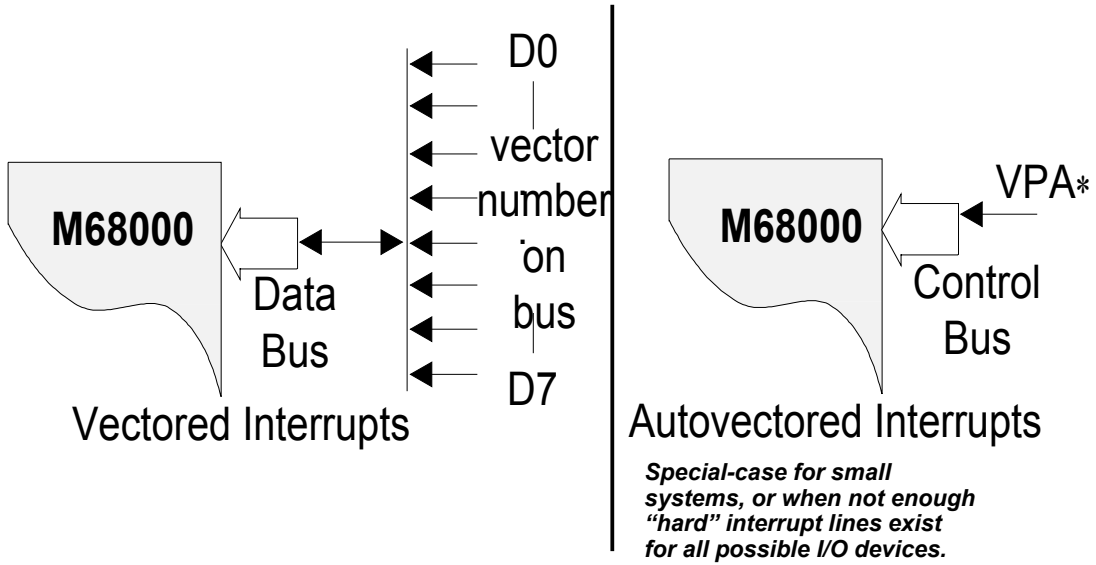
Source Pettus et al. © 1993 Prentice-Hall Publishers



© 2002 Dr. James P. Davis Page 12

# Interrupt Vectors

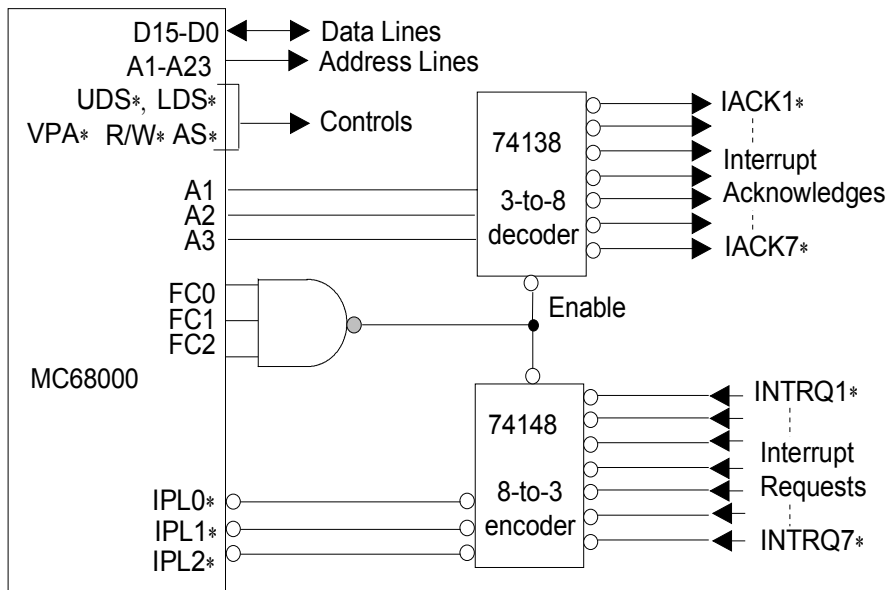
Source Pettus et al. © 1993 Prentice-Hall Publishers



© 2002 Dr. James P. Davis Page 13

# The Control Lines & Multiple Interrupts

Source Pettus et al. © 1993 Prentice-Hall Publishers



© 2002 Dr. James P. Davis Page 14