

Mid-term Examination, November 2003

1) [20] Interrupt Questions

<2 marks> a) Which requires the device to directly specify what device-specific entry in the interrupt vector table to use? (Circle the answer)

(i) Vectored Interrupt

(ii) Autovectored Interrupt

<3 marks> b) If we are using autovectored interrupts and a device with priority 3 causes an interrupt, at what address in the Interrupt Vector Table will the M68K find the starting address of the Interrupt Service Routine? (Show your calculations)

$(3+24)*4 = 108$ (0x6C) is the address

<5 marks> c) Say you design a micro-programmed processor that has 4 distinct steps for each instruction: Instruction Fetch, Instruction Decode & Fetch Operands, Execute, and Writeback. In what step should the micro-program check if there is a pending interrupt? Justify your answer in 1 sentence.

Acceptable answers:

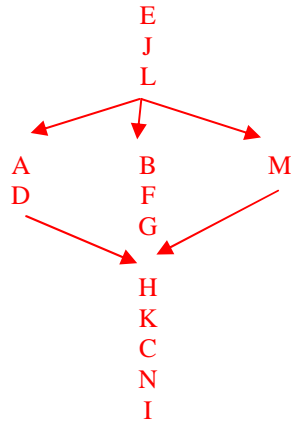
In the writeback step because the processor must determine whether the next step is a fetch or interrupt handler.

At the fetch step address determination, because this is where the usual fetch microaddress will be modified to the interrupt handler microaddress

<10 marks> d)

The following are steps the CPU takes in an interrupt. Put the steps in order and write down the order below. We realize there are several correct answers and getting just one of them will suffice for full marks.

- a) CPU: completes the current instruction
- b) CPU: grants the interrupt for the device
- c) ISR: Restore saved registers from the stack and executes *return from exception* instruction
- d) CPU: must save the state of the cpu before servicing the interrupt (pushing on the stack)
 - i) current PC
 - ii) condition codes (CCR of status register)
- e) Program: CPU configures device for interrupts
- f) Device: sends a vector number to the cpu
- g) CPU: figures out what ISR to run
 - i) uses vector to look up the address of that routine in the *interrupt vector table*
 - ii) sends interrupt acknowledge to the device (IACK)
- h) CPU: Executes ISR
- i) CPU: return to point of execution where interrupted
- j) Program: CPU makes a request from device, and works on something else while waiting
- k) ISR: Saves Registers onto the stack (Callee Save) and execute
- l) Device: when device is ready, it sends an interrupt request to the cpu
- m) CPU: temporarily disables other interrupts
- n) CPU: restores PC and CCR



There are 3 sequences here that can come in any order. So, for example, F must come after B and before G but can come before or after M, A and D. Similarly M can come at any time as long as it is after L and before H.

2) [20] **Processor Architecture**
Consider the two-bus architecture shown, and the following instruction.

`add.w d5, (a3)+`

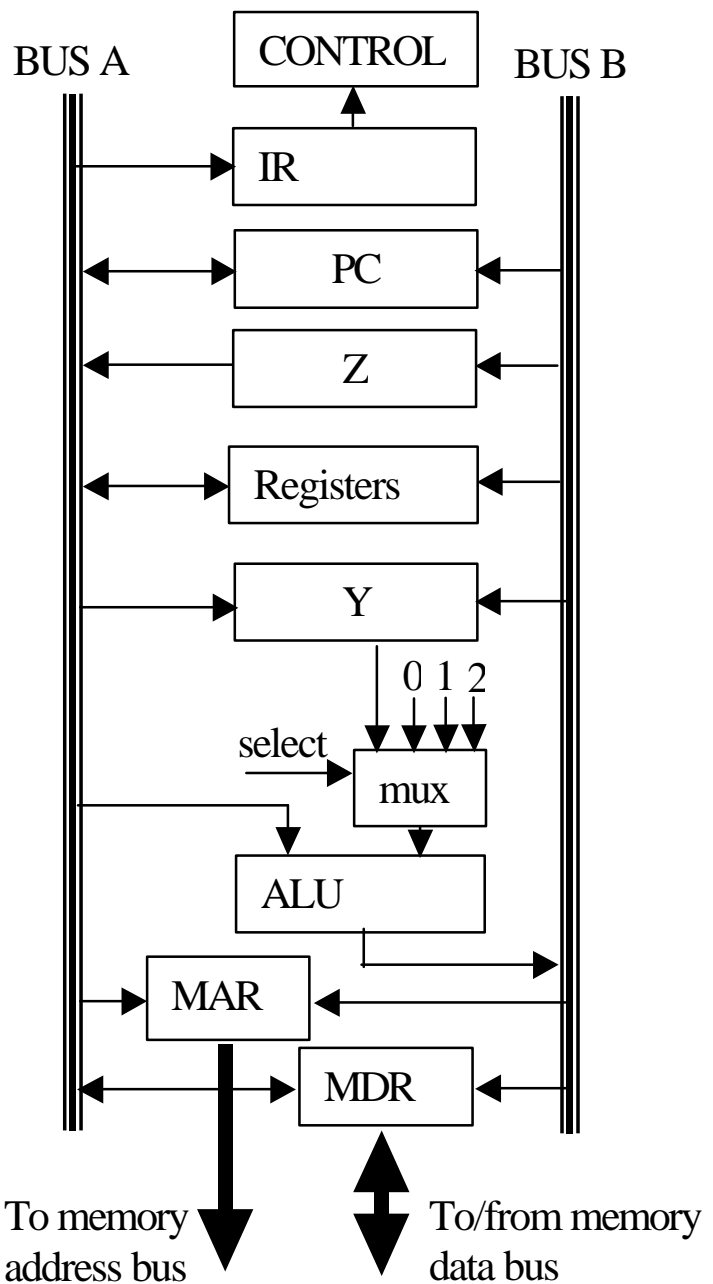
a) Give the register transfer notation for each of the two parts of this instruction (assume that a word is 2 bytes)

<2 marks> $[a3] \leftarrow [[a3]] + [d5]$

<2 marks> $a3 \leftarrow [a3] + 2$

<16 marks> b) Give the steps and control signals for implementing the instruction in the table on the next page. Efficiency will be part of your mark on this question. Use **ONLY** the following control signals:

IRin, PCinA, PCinB, PCout, Zin, Zout, d5inA, d5inB, d5out, a3inA, a3inB, a3out, YinA, YinB,



select Y, select 0, select 1, select2, MARinA, MARinB, MDRinA, MDRinB, MDRout, ALUadd, READ, WRITE, WMFC, END

10 places are provided for your answer – use only the steps you need

SOLUTION NOTES:

- WMFC must appear in a step **AFTER** the corresponding READ or WRITE
 - For a READ, it is most efficient if the WMFC appears in the step where the value returned from memory is first used
- The final WRITE requires a WMFC. There are two correct possibilities for where:
 - In a step after the final write
 - In the very first step (implying that there would have to be a WMFC in the very first step of the implementation of every instruction)
- A missing WMFC costs -1
- Your solutions were graded for efficiency:
 - Using steps 7 or 8 costs -1
 - Using steps 9 or 10 costs an additional -1
- Of course, the rest of your solution was graded for accuracy and completeness

Sample solution1 (worth 16---full marks); note that it is very efficient

Step 1	WMFC, PCout, MARinA, READ, select 2, ALUadd, PCinB
Step 2	WMFC, MDRout, IRin
Step 3	a3out, MARinA, READ, select 2, ALUadd, a3inB
Step 4	d5out, YinA
Step 5	WMFC, select Y, MDRout, ALUadd, MDRinB, WRITE, END

Sample solution2 (worth 16); slightly less efficient, since the WMFC is in step 4 immediately after the READ in step 3 (having extra steps between a READ and it's WMFC buys the memory more time to return the value)

Step 1	WMFC, PCout, MARinA, READ, select 2, ALUadd, PCinB
Step 2	WMFC, MDRout, IRin
Step 3	a3out, MARinA, READ, select 2, ALUadd, a3inB
Step 4	WMFC, MDRout, YinA
Step 5	d5out, select Y, ALUadd, MDRinB, WRITE, END

Sample solution3: (worth 14); correct but inefficient: -1 for using steps 7/8, -1 for steps 9/10

Step 1	PCout, MARinA, READ
Step 2	PCout, select 2, ALUadd, PCinB
Step 3	WMFC, MDRout, IRin
Step 4	a3out, MARinA, READ
Step 5	a3out, select 2, ALUadd, a3inB
Step 6	WMFC, MDRout, YinA
Step 7	d5out, select Y, ALUadd, MDRinB,
Step 8	WRITE
Step 9	WMFC, END

3) [20] **Attached Devices**

<3 marks>

- a) A processor uses a serial link to communicate with a keyboard for word processing. A typist using this keyboard can type at rates peaking at 120 words per minute, where a word is 6 characters (including spaces and punctuation). The characters will be transmitted from the keyboard in 8-bit ASCII with one stop bit and no parity. Only consider these and no special characters.
 If the processor allows selections of baud rates from the table at the right, circle the minimum baud rate that can be selected and still have the keyboard work for all typists? Show your calculations that you used to determine your result (these will be marked).

Baud Rates
300
600
1200
2400
4800
9600
19200

120 wpm → 120*6/60 cps = 12 cps

Each character requires 10 bits sent (8 bits for the character, 1 start, 1 stop)

Thus the minimum baud rate is 12*10=120 baud

Thus 300 baud is the setting to use.

Note that this is a fast typist!

<2 marks>

- b) The programmer writes setup and polling service routines based on a minimum baud rate, but then finds that the keyboard will only interface at 19.2 Kbaud. Will the polling service routine have to change? Briefly: Why or why not?
 No it won't have to change. The interrupts will not come more frequently, just slightly faster after every keystroke. This is not something that necessitates change to the polling service routine.
 Note that the setup routine would have to change to match the different baud rate.

c) Attaching a watchdog.

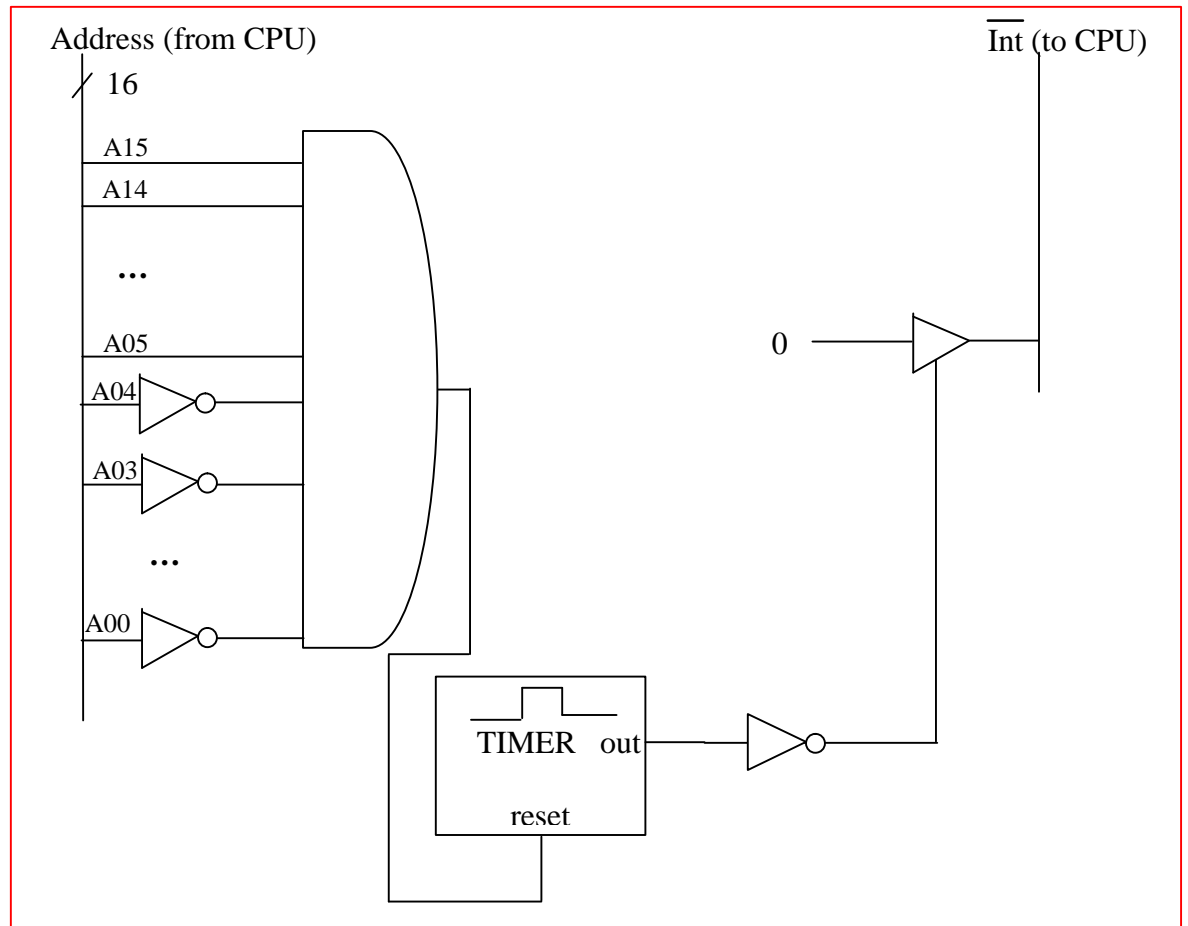
A simple microprocessor is to have a watchdog timer installed. This timer will interrupt the processor unless it is constantly reset by the processor which it would normally do. This interrupt would happen if the processor went 'off the rails' because of a software problem, a power spike or something like that.

A circuit diagram is started below. The timer in the diagram gives a single positive-going pulse which can be extended by 'retriggering' using the reset line. Some information about the timer and use are given on the diagrams on the next page.

The interrupt line is shared by other devices and is active when low.

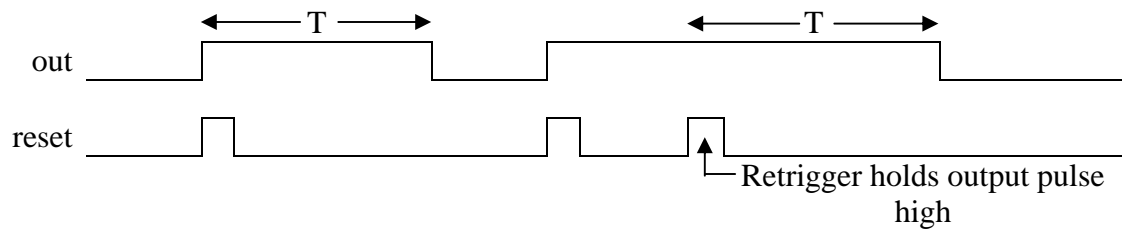
In the scheme to be used, the processor will reset the timer (and thus avoid the interrupt) by accessing a specific address, \$FFE0, faster than the pulse time of the timer, thus restarting the pulse without an interrupt being generated.

- <10 marks> (i) Attach the timer to the 16-bit address bus and to the interrupt line so it will respond as required using supporting logic. For full marks, use ands /ors / nots and other gate-level devices only.

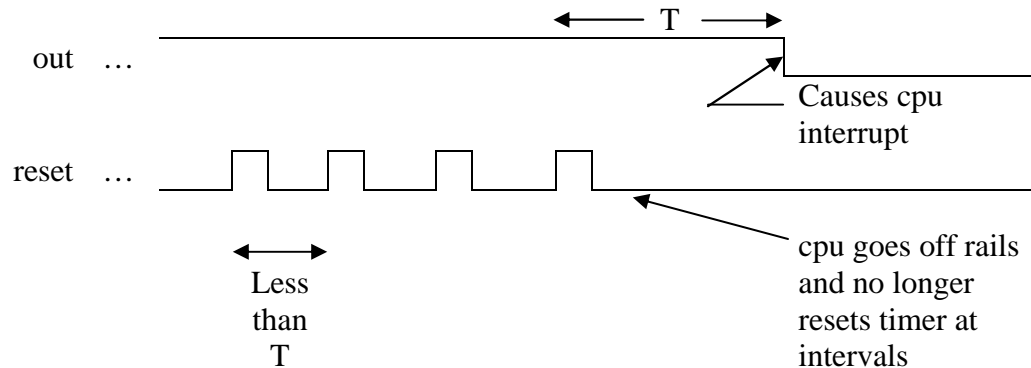


Timer: Relationship between reset input and the output:

$T = \text{pulse time}$



Use of watchdog:



- <5 marks> (ii) Address lines do not change instantaneously or absolutely at the same time. They may drift when not being driven by some source. These factors could cause accidental triggering of the watchdog if the address lines even momentarily have the value \$FFE0. Describe briefly or show a method that might be used to greatly reduce or to eliminate this problem.

Possibilities:

- use MRDY or processor clock or bus clock to indicate when the address is valid
- use consecutive writes to 2 different (best non-consecutive) addresses to reset the timer
- write a specific value of data to lower the probability of problem.
- use a delay and and gate or other debounce circuit (see below)

