Chris Thomas

Set #2 (Due Wed Oct 19)

1) The PIC32MZEC processor's EIC currently supports a possible 191 interrupt vectors. If we wanted to provide support for a possible 256 vectors,
a) how many IFSx registers and IECx registers would be needed? (Note that the current number of IFSx and IECx registers could support 192 vectors.)

Both the IFSx and IECx registers are 32 bits wide. So, each can handle 32 vectors. 256/32 = 8 therefore we need 8 of each register to handle 256 vectors.

b) how many IPCx registers would be needed? Assume that each IPC register supports as many vectors as it does in the current architecture.

Since the IPCx registers support 4 vectors each to have 256 vectors we need to do 256/4 = 64 so we need 64 IPCx registers.

2) Interrupt Mode

a) What processor core interrupt mode is the PIC32MZ core in when it first powers up? (Note that we are not talking about the vector address computations modes (computed offset vs variable offset mode) in this question. The vector address computation mode is determined by the processor's hardware configuration and cannot be changed.)

When the PIC32 processor (Release 2 MIPS M4K) first boots, it is in "Interrupt Compatibility Mode". In this mode interrupts are handled as they were in a Release 1 MIPS processor meaning that individual interrupts are not vectored, and all interrupts are handled in a single service routine.

b) What register bit settings have to be changed (and to what values) for the PIC32MZ core to switch from the mode it powers up in to EIC mode?

Cause.IV = 1
IntCtl.VS ≠ 0

Status.BEV = 0

3) Interrupts

a) Assume that a PIC32 processor is in EIC multi-vector mode, that the `Ebase` register currently contains `0x9FC0.1000` and that `Intctl.VS = 0x04`. If computed offsets are used, at what address must the first instruction of the service routine for vector 27 be stored?

Ebase = 0x9FC0.1000
Intctl.VS = 0x04
Computed Offset Vector Address = (Vector# x Vector Spacing + 0x200) + EBase

0x1B * 0x04(32) * 0x200 + 0x9FC0.1000 = 0x9FC01F80

b) If variable offsets are used instead, and we wanted the address of the first instruction of the service routine for vector 27 to be the same as you found in part a, what would have to be done? Assume the same register settings given in part a.

Variable Offset Vector Address = EBase + Vector Offset (OFFx)
Variable Offset = 0x9FC01F80 - 0x9FC0.1000 = 0xF80
We have to set the variable offset to be 0xF80.

c) Why will an interrupt whose group priority is 0 never be serviced?

The user-selectable priority levels range from 1 (lowest) to 7 (highest). If an interrupt priority is set to zero, the interrupt vector is disabled for both interrupt and wake-up purposes. Interrupt vectors with a higher priority level preempt lower priority interrupts.

d) Assume that a PIC32MZ processor is properly configured for EIC mode (see Q 2.b). What other register bit settings are required for an interrupt to be serviced?

IE = 1
EXL = 0
ERL = 0
DM = 0

e) What role does the sub priority and the IRQ natural order play in deciding which of several simultaneous interrupt requests will be served?

If two interrupts with the same priority are pending, the interrupt with the highest subpriority will be handled first. The natural priority is a fixed priority scheme, where the highest natural priority starts at the lowest interrupt vector, meaning that interrupt vector 0 is the highest natural priority.

f) Can an interrupt with group priority 3 and sub-priority 2 interrupt a service routine with group priority 3 and sub-priority 1? Why or why not?

The subpriority will not cause preemption of an interrupt in the same priority. The subpriority is strictly for if two interrupts with the same priority are pending.

4) Assume that a PIC32 processor is in EIC mode, that the `Ebase` register currently contains `0x9FC0.1000` and that `Intctl.VS = 0x04`.

a) What is the address of the service routine for an NMI?

0xBFC0.0000 (Found in datasheet)

b) What is the address of the service routine for an Overflow exception?

EBASE+0x180 = 0x9FC0.1000 + 0x180 = 0x9FC01180

c) What is the address of the service routine for a Reserved Instruction (aka Undefined Instruction) exception?

EBASE+0x180 = 0x9FC0.1000 + 0x180 = 0x9FC01180

5) Section 8.6.2 (and section 8.4) describes how, in Single vector mode, the EIC forces all IRQs to use vector number 0. Equation 8.3 gives the formula for the EIC Single Vector mode ISR address for both computed offset and variable offset processors.
a) Show how the formula for computed offsets (the corrected equation 8.1) would give the same result as equation 8.3 if the EIC was in single vector mode.

Single Vector Address = EBase + 0x200

Computed Offset Vector Address = (Vector# * Vector Spacing + 0x200) + Ebase

If Vector# = 0 then 0 * Vector Spacing = 0 which makes the equations just EBase + 0x200.

b) The documentation is not precisely clear as to how the single vector address given by equation 8.3 is arrived at in processors that use variable offset. Why might it be necessary to set OFF000 to 0x200 in single vector mode on a variable offset processor in order to equation 8-3 to be valid?

Variable Offset Vector Address = EBase + Vector Offset (OFFx) so setting it to 0x200 gets us to the formula of 8-3 which is EBase + 0x200.

6) MIPS documentation informs us that no part of the vector offset that is added to Ebase (last step in the vector address calculation) may have any non-zero bits that are

the same or more significant than the least significant non-zero bit in Ebase. If this condition is not met, operation of the processor is undefined.
a) What does this suggest about how this last addition in the vector address calculation might actually be implemented in the processor hardware?

Address calculation uses OR gates.

b) Given that the Ebase register currently contains 0x9FC0.4000 and that Intctl.VS = 0x04, what is the maximum vector that can be handled in computed offset mode? Please show your work.

3FFF >= (Vector# *0x80 + 0x200)
3DFF / 0x80 = Vector# = 7B

c) What is the maximum value that can be stored in a PIC32 OFFxx register for an active vector if Ebase currently contains 0x9FC0.4000 ? (Assume we are not in microMIPS mode.)

0x7B * 0x80 + 0x200 = 0x3F80

d) What is the maximum value that can be stored in a PIC32 OFFxx register for an active vector if Ebase currently contains 0x9FC8.0000 ? (Assume we are not in microMIPS mode.)

0x7FFFF >= Vec * 0x80 + 0x200
0x7FFFF – 0x200 / 0x80 = FFB

e) Given that the legal values of Intctl.VS only have one bit asserted, what does this tell you about how the Vec# × vector spacing part of the computed vector address calculation is probably implemented in the processor hardware?

All of our legal values are multiples of two, so it could be implemented using shifters

f) How do your answers to parts c or d change if the processor is in microMIPS mode?

microMIPS compresses instructions to 16 bits.