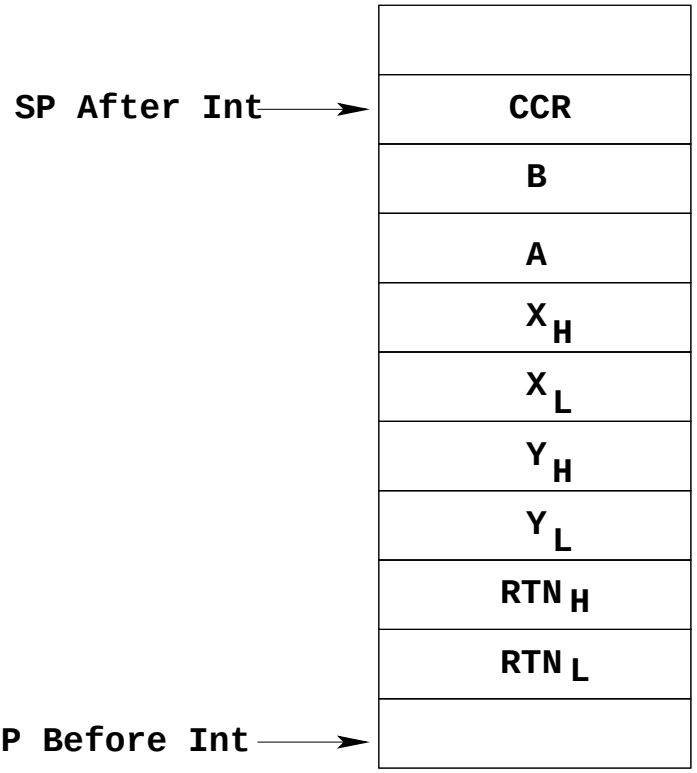
EXCEPTIONS ON THE HC12

- Exceptions are the way a processor responds to things other than the normal sequence of instructions in memory.
- Exceptions consist of such things as Reset and Interrupts.
- Interrupts allow a processor to respond to an event without constantly polling to see whether the event has occurred.
- On the HC12 some interrupts cannot be masked — these are the Unimplemented Instruction Trap and the Software Interrupt (`SWI` instruction).
- XIRQ interrupt is masked with the X bit of the Condition Code Register. Once the X bit is cleared to enable the XIRQ interrupt, it cannot be set to disable it.
  - The XIRQ interrupt is for external events such as power fail which must be responed to.
- The rest of the HC12 interrupts are masked with the I bit of the CCR.
  - All these other interrupts are also masked with a specific interrupt mask. For example, the Timer Overflow Interrupt is masked with the TOI bit of the TMSK2 register.
  - This allows you to enable any of these other interrupts you want.
  - The I bit can be set to 1 to disable all of these interrupts if needed.

USING INTERRUPTS ON THE HC12

What happens when the HC12 receives an unmasked interrupt?

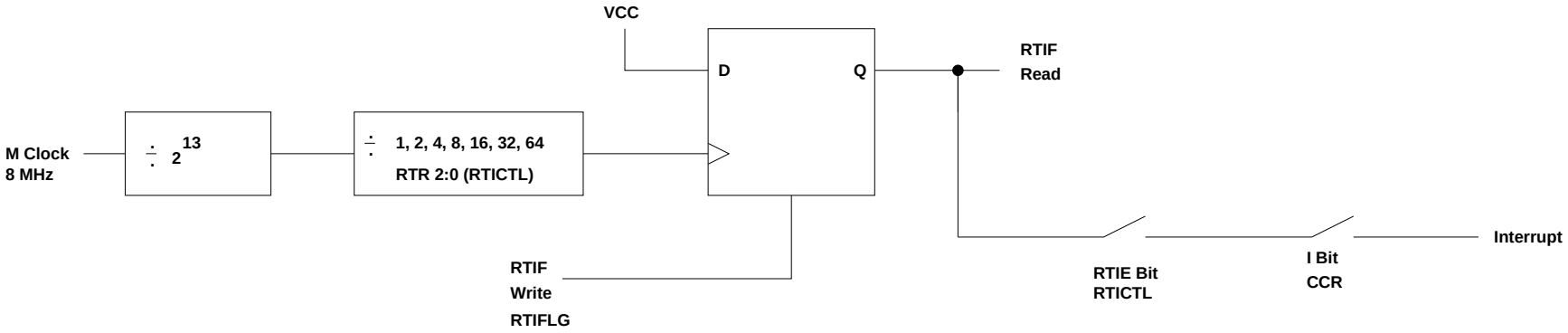
1. Finish current instruction
2. Clear instruction queue
3. Calculate return address
4. Push Return Address, Y, X, A, B, CCR onto stack (SP is decremented by 9)



5. Set I bit of CCR
6. If XIRQ interrupt, set X bit of CCR
7. Load Program Counter from interrupt vector for highest priority interrupt which is pending

The Real Time Interrupt

- Like the Timer Overflow Interrupt, the Real Time Interrupt allows you to interrupt the processor at a regular interval.



- The specific interrupt mask for the Real Time Interrupt is the RTIE bit of the RTICTL register.
- When the Real Time Interrupt occurs, the RTIF bit of the RTIFLG register is set.
  - To clear the Real Time Interrupt write a 1 to the RTIF bit of the RTIFLG register.
- The interrupt rate is set by the RTR 2:0 bits of the RTICTL register.

RTIE	RSWAI	RSBCK	0	RTBYP	**RTR2**	**RTR1**	**RTR0**	**0x0014**	**RTICTL**

RTIF	0	0	0	0	0	0	0	**0x0015**	**RTIFLG**

- Here is the interrupt rate for an HC12 with an 8 MHz M-Clock:

RTR 2:0	Rate
000	Off
001	1.024 ms
010	2.048 ms
011	4.096 ms
100	8.192 ms
101	16.384 ms
110	32.768 ms
111	65.536 ms

- To use the Real Time Interrupt, set the rate by writing to the RTR 2:0 bits of RTICTL, and enable the interrupt by setting the RTIE bit of the RTICTL register.
  - In the Real Time Interrupt ISR, you need to clear the RTIF flag by writing a 1 to the RTIF bit of the RTIFLG register.

- Here is a C program which uses the Real Time Interrupt:

```
#include "hc12b32.h"

main()
{
    DDRA = 0xff;
    PORTA = 0;

    RTICTL = 0x85;   /* Enable RTI, set rate to 16.384 ms */
    enable();
    while (1)
    {
        _asm("wai");   /* Do nothing -- wait for interrupt */
    }
}

@interrupt void rti_isr(void)
{
    PORTA = PORTA  + 1;
    RTIFLG = 0x80;
}
```

- Note that in the above program, the do-nothing loop has the instruction

```
asm("_wai");   /* Do nothing -- wai for interrupt */
```

  The assembly-language instruction `WAI` (Wait for Interrrupt) stacks the registers and puts the HC12 into a low-power mode until an interrupt occurs.

- This allows the HC12 to get into the ISR more quickly (because the time needed for pushing the registers on the stack has already been done), and lowers the power consumption of the HC12 (because it doesn't have to execute a continuous loop while waiting for the interrupt).