**Timing Diagrams**

* Either a LOAD or STORE instruction
* Both have a common CLK signal
* Cycle 1 ROM is ALWAYS highk

**Table

Description automatically generated**

**Parallel Port Programming** **(C)**

* There are two types of parallel port logic:

1. **Central Decoding Logic**
   1. Simplifies the decoding logic, there is a common decoder output, which is part of the Boolean expression
   2. Will have and the rest of the Boolean expression
2. **No Central Decoding**
   1. Directly convert the register address to binary and then apply

**Interrupts and Hardware Programming (NIOS II):**

* **Setting Up for Interrupts**
  + First, we need to setup the ***ienable*** and ***ipending*** registers.
    - Given an assigned bit, use a temporary register, move the bit pattern into this temporary register
    - Then use to ***wrctl*** command to write the value to the respective registers
* **Checking for Interrupts**
  + First, we use the ***rdctl*** command, we read the ***ipending*** contents into a temporary register ***et***
    - And ***et*** with the assigned bit pattern
    - BRANCH IF the result is 0 (r0)
    - Else call the ISR

Remember, ISR uses ERET compared to RET.

Know the control registers

* We WRITE (***wrctl***) to: ***ienable, status, estatus***
* We READ (***rdctl***) from: ***ipending***

**Embedded Application Programming**

* To figure out the time:
  + Multiply the clock frequency (usually MHz) with the timer interrupt interval
    - Convert this number of cycles to hex
    - Split this 8 digit hex number (32 bits) into two 4 digit hex numbers (16 bits) and store them in pointers for the low and high value of the number of cycles respectively.
* Setting up the timer
  + Set the timer status to 0
  + Set the timer control to 7, enabling STOP, START, CONT (7 = 0b111)
* Enable interrupts
  + Write 1 to the first bit (bit 0) of the IENABLE and STATUS (0 = 0b1)
* Setting up the interrupt handler
  + Read the value of the IPENDING register into a variable
  + Check

1- Define headers, pointers and global variables  
- Use the chart and the table  
- Read the question thoroughly, learn how to identify global variables needed (led\_pattern, seconds, one\_minute, etc.)  
2- Initialization Code  
- software variables- set the global variables to appropriate start values  
- set up the interfaces- timer, output, input ports, other devices  
- set up ienable in processor- NIOS2\_WRITE\_IENABLE  
-set up status in processor- NIOS2\_WRITE\_STATUS  
3. Interrupt Handler  
- create unsigned int ipending- NIOS2\_READ\_IPENDING  
-if (ipending &0x1)  
-proceed with specification  
-create a loop that raises the software flag once condition is met  
4. Main  
-define local variables  
-initialize variables  
-infinite loop  
-check the software flag condition  
-clear it  
-proceed with specifications