Examen Tercer Parcial **ESTRUCTURA DE COMPUTADORES** **19 Mayo 2014**

Apellidos y Nombre **DNI** **Grupo**

SOLUCIÓN		

1. (**3 puntos**) Se dispone de un computador basado en MIPS R2000 que se haya conectado a dos periféricos: un reloj de tiempo real y un termómetro. La definición de los interfaces de ambos periféricos es la siguiente:

Reloj de tiempo real cuya dirección base (DB_R) es 0xFFFF0000. El adaptador dispone de 4 registros de 8 bits:

- **Hora** (BD_R): Registro de solo lectura que contiene la hora actual (0..23)
- **Minutos** (DB_R+1): Registro de solo lectura que contiene los minutos actuales (0..59)
- **Segundos** (DB_R+2): Registro de solo lectura que contiene los segundos actuales (0..59)
- **Estado** (DB_R+3): Registro de lectura/escritura con un único bit significativo:
  - **R:** READY (bit 0). Este bit se activa a 1 con cada variación de segundos. Los registros de Hora, Minutos y Segundos son debidamente actualizados por el hardware. El bit R debe ser puesto a cero por software a fin de poder detectar el siguiente cambio del reloj.

Termómetro cuya dirección base (DB_T) es 0xFFFF0200. Se trata de un dispositivo siempre preparado y que dispone de un único registro de 8 bits:

- **Temp** (BD_T): Registro de solo lectura actualizado por hardware. Contiene la temperatura actual representada por un valor entero de 8 bits (-128..127).

Escriba un programa capaz de adquirir 1000 muestras de temperatura. El programa debe almacenar una medida de temperatura cada vez que haya un cambio en el registro Minutos del Reloj. Si el sistema se inicia en tiempo 0:0:0, entonces deberíamos tomar muestras de temperatura en los tiempo 0:1:0, 0:2:0, 0:3:0, etc. Obsérvese que el Reloj cambia cada segundo, pero las medidas de temperatura deben hacerse sólo cuando el valor en el registro Segundos es cero. Los periféricos no están preparados para interrumpir, por lo que la sincronización deberá hacerse por consulta de estado. Para almacenar las 1000 medidas de temperatura, se define la siguiente estructura de datos:

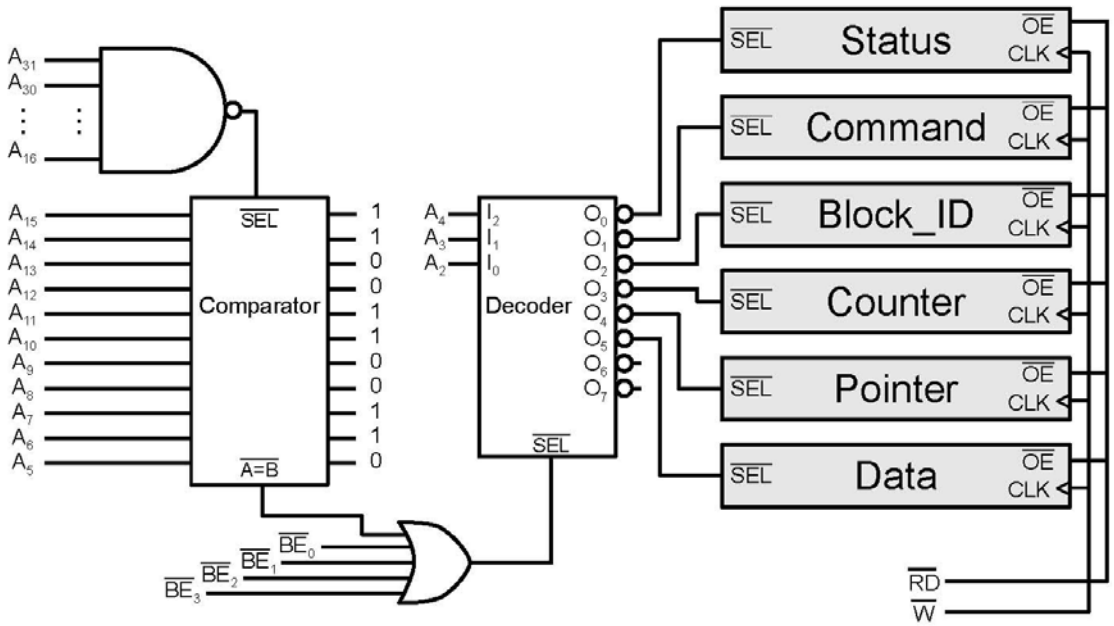
```
      .data 0x10000000
Temps:     .space     1000 #Almacena las muestras de temperatura
```

Los valores de temperatura deberán almacenarse secuencialmente, uno tras otro correspondiendo a minutos consecutivos. Se asume que el programa se inicia en modo Kernel, por lo que se dispone de acceso a todo el espacio de direccionamiento del procesador. Deberá hacer uso de los siguientes registros:

- $t0: Contendrá dirección base del Reloj (DB_R)
- $t1: Contendrá dirección base del Termómetro (DB_T)
- $t2: Se utilizará como contador de las 1000 iteraciones requeridas
- $t3: Se utilizará como puntero para recorrer el array Temps
- $t4: Puede ser usado para otros propósitos

```
        .data 0x10000000
Temps: .space 1000          # To store the temperature log
        .ktext
       la $t0, 0xFFFF0000   # Clock's base address BA_C
       la $t1, 0xFFFF0020   # Thermometer's base address BA_T
       li $t2, 1000         # Size of log
       la $t3, Temps        # Pointer to Temps[0]
loop:  lb $t4, 3($t0)       # Read Status register
       andi $t4, $t4, 1     # Mask out the Ready bit
       beqz $t4, loop       # Polling for Ready bit = 1 in Status
       sb $zero, 3($t0)      # Reset Ready bit
       lb $t4, 2($t0)       # Read Sec
       bnez $t4, loop        # Ignore seconds values other than 0
       lb $t4, 0($t1)       # Read temperature from thermometer
       sb $t4, 0($t3)        # Store temperature to Temps
       addi $t3, $t3, 1     # Update Temps pointer
       addi $t2, $t2, -1    # Decrease counter
       bnez $t2, loop        # Iterate until counter becomes 0
       .end
```

2. (**2 puntos**) La Figura 1 muestra el adaptador de cierto periférico de bloques con capacidad de DMA, el cual se halla conectado a un procesador MIPS R2000.



Obsérvese que todas las líneas de dirección desde A31 a A16 están conectadas directamente a una puerta NAND (sin invertir las entradas). Los seis registros son de 32 bits.

2.1 - ¿Cuál es la dirección base (DB) del adaptador?

DB = 0xFFFFCCC0

2.2 - Indíquese la dirección de cada uno de los seis registros en la forma <DB+desplazamiento>

Status = BA

Command = BA + 4

Block_ID = BA + 8

Counter = BA + 12

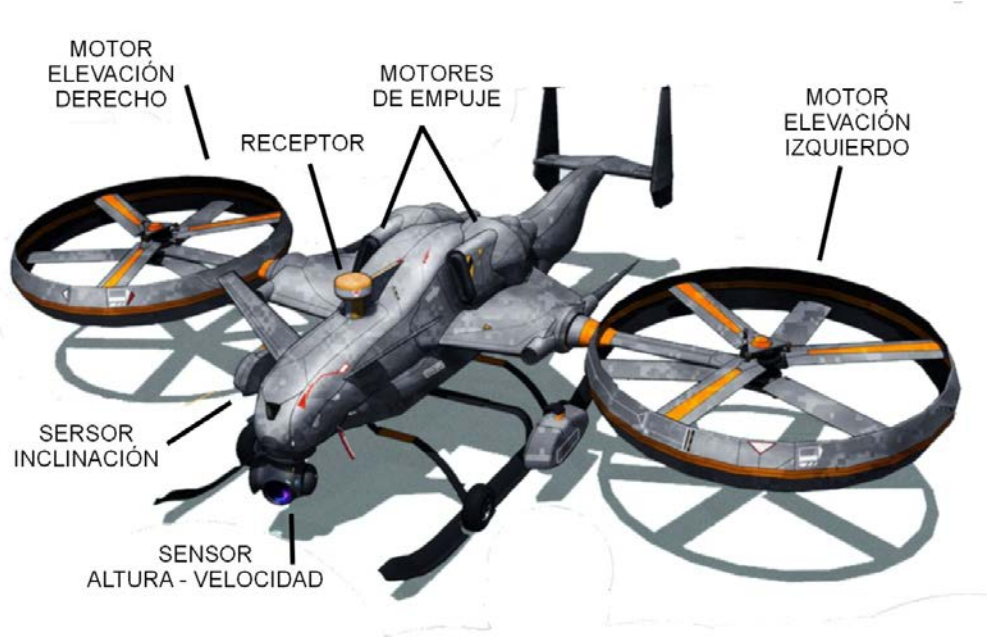
Pointer = BA + 16

Data = BA + 20

2.3 - Escriba el código para programar una transferencia por DMA desde el periférico a memoria (transferencia de lectura). El inicio de la operación de DMA es ordenada poniendo a '1' los bits 0 y 1 del registro <Command>. Se desea leer el bloque cuyo identificador es 0x01234567. El tamaño del bloque es de 512 bytes, si bien en cada ciclo se transfiere una palabra de 32 bits. El bloque debe almacenarse en memoria en la dirección etiquetada como ***Mem_Block***.

```
la $t0,0xFFFFCCC0
li $t1,0x01234567 # Block_ID
sw $t1,8($t0)
la $t1,Mem_Block # Pointer
sw $t1,0x10($t0)
li $t1,128 # Count = 512 / 4
sw $t1,0xC($t0)
li $t1,3 # Command
sw $t1,4($t0)
```

3. (**5 puntos**) La figura 1 muestra el *drone* ***Predator-EC*** para vigilancia de exámenes de EC. El aeromodelo consta de dos motores de elevación (izquierdo y derecho), dos motores de empuje (izquierdo y derecho), un sensor de altura y velocidad, un sensor de inclinación y un receptor Wi-fi. El conjunto se controla mediante un microcontrolador de arquitectura MIPS R2000.



El sistema informático se compone de los siguientes periféricos:

CONTROL DE MOTORES: Dirección Base DB=0xFFFF0000 Registros de 8 bits Lectura/Escritura

Registro	**Dirección**	**Descripción**	**Valores**
Motor_empuje_I	DB	Potencia motor empuje izquierdo (8 bits)	0 = parado 1:200 Potencia de empuje
Motor_empuje_D	DB+1	Potencia motor empuje derecho (8 bits)	
Motor_elevacion_I	DB+2	Potencia motor elevación izquierdo (8 bits)	0 = parado < 100 bajar 100 = estabilizado 101..200 subir
Motor_elevacion_D	DB+3	Potencia motor elevación derecho (8 bits)	

La aeronave despega verticalmente poniendo los motores de elevación a potencia mayor que 100. Si se fija la potencia de estos motores a valor 100 la nave se sustenta en el aire (ni sube un baja). Para avanzar se debe dar potencia a los dos motores de empuje simultáneamente. Para girar la nave a izquierdas hay que aumentar la potencia del motor de empuje derecho en un valor de 10. Si en el giro se desea también una inclinación lateral hay que aumentar la potencia del motor de elevación derecho en un valor de 10. Lo mismo para el giro a derechas, actuando sobre los motores izquierdos. Este periférico es de E/S directa, es decir no necesita sincronización y simplemente escribiendo los valores en los registros cambia la potencia de giro de los motores correspondientes.

SENSOR DE ALTURA-VELOCIDAD: Dirección Base DB=0xFFFF0010 Interrupción ***Int_1**

Registro	**Dirección**	**Descripción**	**Valores**
Control /Estado (8 bits Lectura/Escritura)	DB	Bit 0 = ON/OFF (marcha/paro) Bit 1 = IE (habilitación de interrupción) Bit 2 = R (Preparado – lectura) Bit 3 = C (cancelación de interrupción) Bits 4..7 = FA (frecuencia de adquisición)	FA = 0 : No adquirir FA = 1::15 Hz R=1 con cada adquisición de altura y velocidad. Si IE=1 se activa la *Int_1.
Altura (16 bits Lectura)	DB+8	Altura de vuelo en centímetros	
Velocidad (16 bits Lectura)	DB+12	Velocidad del aire en cm/s	

Cuando se activa el sensor (ON/OFF=1) éste adquiere medidas de altura/velocidad de forma repetitiva a la frecuencia indicada en FA. Si el bit IE = 1 se produce la interrupción *Int_1 con cada adquisición. La cancelación se produce al hacer C=1 (hay que dejar el resto de bits como estaban).

SENSOR DE INCLINACIÓN: Dirección Base DB=0xFFFF0020 Interrupción ***Int_2**

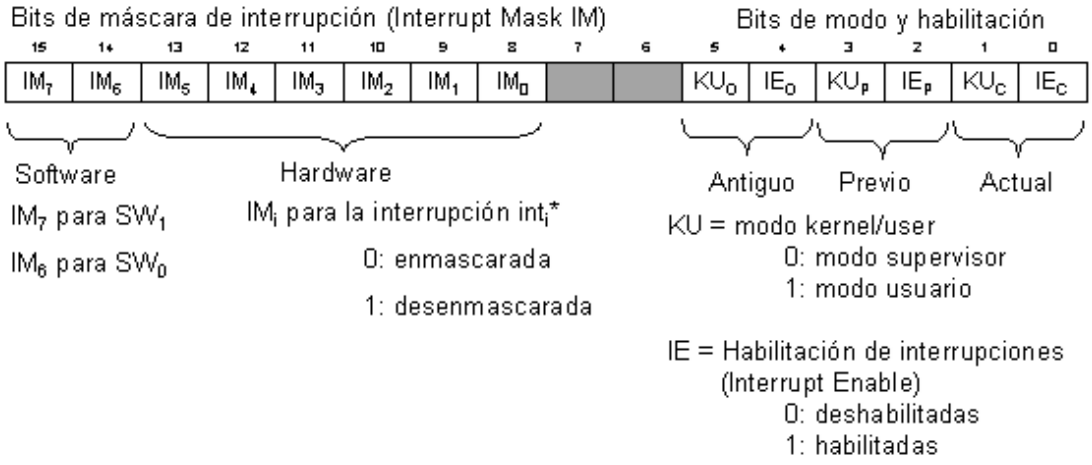
Registro	**Dirección**	**Descripción**	**Valores**
Control/Estado (8 bits Lectura/Escritura)	DB	Bit 0 = MODO (0:NORMAL 1:ESTABILIZACIÓN) Bit 1 = IE (habilitación de interrupción) Bit 7 = R (Preparado –solo lectura)	En ESTABILIZACIÓN R=1 cada vez que la inclinación varía en un grado. Si IE=1 se activa la *Int_2.
Inclinación (8 bits-Lectura)	DB+4	Ang : Ángulo de inclinación en grados (-90..90)	Ang = 0 – horizontal Ang > 0 – inclinado a derecha Ang < 0 – inclinado a izquierda

En modo NORMAL el sensor suministra el valor de la inclinación lateral del aparato en el registro <***Inclinacion***>, de forma continua. En modo ESTABILIZACIÓN, el sensor emite una interrupción sólo cuando hay un cambio de un grado en el ángulo de inclinación. La cancelación de la interrupción se produce al leer el registro <***Inclinacion***>.

Se pide:

3.1 (**1 punto**) Escriba un fragmento de código para inicialización del sistema, que debe: poner la potencia de todos los motores a cero, activar el sensor de altura-velocidad para que capture los valores a 10 Hz y emita la interrupción correspondiente. También debe activar el sensor de inclinación en modo NORMAL y SIN interrupciones. Inicializar a cero las variables del *kernel* que se indican a continuación. Por último, debe pasar a modo usuario con las interrupciones habilitadas y las interrupciones **Int_1** e **Int_2** desenmascaradas.

```
   .kdata 0x00000000
Potencia_Empuje_Nominal:        .byte 0   # Potencia motores
Potencia_Elevacion_Nominal:     .byte 0

Altura:           .half 0     # Variables de navegación
Velocidad:        .half 0

Modo:             .byte 0   # Modo Sensor Inclinación = NORMAL
Inclinacion:      .byte 0

   .ktext 0x00400000
 __start:    la $t0, 0xFFFF0000     # DB de Control Motores
             sb $zero, 0($t0) # Potencia Empuje nominal = 0
             sb $zero, 1($t0)
             sb $zero, 2($t0) # Potencia Elevación nominal = 0
            sb $zero, 3($t0)
            sb $zero, Potencia_Empuje_Nominal
             sb $zero, Potencia_Elevacion_Nominal


             la $t0, 0xFFFF0010     # Db de Sensor Altura-Velocidad
             li $t1, 0xA3           # ON/OFF=1, IE=1, FA = 10
             sb $t1, 0($t0)
             sb $zero, Altura
             sb $zero, Velocidad

            la $t0, 0xFFFF0020     # Db de Sensor Inclinación
             sb $zero, 0($t0)       # MODO = NORMAL, IE = 0
             sb $zero, Modo

     # En el procesador
             li $t1, 0x0603         # IM1 = 1,IM2=1,IE = 1, K/U = 1 modo usuario
             mtc0 $t1, $12
```



3.2 (**1.5 puntos**) Para las operaciones de vuelo se dispone de las siguientes funciones del sistema (syscalls)

Nombre	Función	Argumentos entrada	Argumentos salida
Go_Forward:	**$v0 = 100**	**$a0 = Potencia de avance**	
Ascend:	**$v0 = 101**	**$a0 = Incremento potencia elevación**	----------
Descend:	**$v0 = 102**	**$a0 = Decremento potencia elevación**	----------
Acquire:	**$v0 = 103**	----------	**$a0= Altura $a1=Velocidad**
Stabilize	**$v0 = 104**	**$a0 = ON: 1 / OFF: 0**	----------
Turn_left:	**$v0 = 105**	**$a0 = Incremento potencia giro**	
Turn_rigth:	**$v0 = 106**	**$a0 = Incremento potencia giro**	

Escriba sólo el código de las funciones que se indican a continuación:

NOTA: Se pueden usar los registros $t0, $t1 y $t2 en el manejador de excepciones.

- **Go_Forward ($a0=Potencia avance)** Activa los motores de avance a la potencia nominal indicada en $a0. Actualiza la variable correspondiente del kernel.

```
Go_Forward:    la $t0, 0xFFFF0000    # DB de Control Motores
               sb $a0, 0($t0)
               sb $a0, 1($t0)
               sb $a0, Potencia_Empuje_Nominal









           b retexc
```

- **Turn_left ($a0=Incremento de potencia)** Incrementa la potencia del motor de avance derecho para girar a la izquierda dejando el motor izquierdo a su potencia nominal actual. Si la velocidad de la nave es mayor que cero también inclina la nave actuando apropiadamente sobre los motores de elevación. El incremento de potencia, en ambos casos, se indica en $a0.

```
Turn_left:   la $t0, 0xFFFF0000      # DB de Control Motores
             lb $t1, Potencia_Empuje_Nominal
             sb $t1, 0($t0)    # Motor izquierdo a potencia nominal
             addu $t1, $t1, $a0
             sb $t1, 1($t0)    # motor derecho a potencia nominal + $a0
             lh $t1, Velocidad
             beqz $t1, retexc

              lb $t1, Potencia_Elevacion_Nominal
             sb $t1, 2($t0)    # Motor izquierdo de elevación a potencia nominal
             addu $t1, $t1, $a0
             sb $t1, 3($t0)    # motor derecho elevac. a potencia nominal + $a0









             b retexc
```

3.3 (**1 punto**) Escriba el código de la rutina de servicio de la interrupción **Int_1** del sensor de Altura-Velocidad. Esta rutina debe tomar las medidas correspondientes del sensor y actualizar las variables del kernel.

```
Int_1:  la $t0, 0xFFFF0010    # DB de Sensor Altura-Velocidad
        lh $t1, 8($t0)
        sh $t1, Altura        # Actualiza Altura
        lh $t1, 12($t0)
        sh $t1, Velocidad           # Actualiza Velocidad
        lb $t1, 0($t0)
         ori $t1, $t1, 0x08
         sb $t1, 0($t0)       # Cancelación *Int_1









       b retexc
```

3.4 (**1.5 puntos**) La función de sistema ***Stabilize ($a0: ON/OFF)*** permite entrar en un modo de vuelo estabilizado. En este modo la nave se detiene en el aire al fijar la velocidad de los motores de elevación a 100 (valor de sustentación). Además, se activa el sensor de inclinación en modo ESTABILIZACIÓN de forma que cualquier inclinación lateral de la nave, por ejemplo debido al viento, hace que se active la interrupción **Int_2** que permite corregir esta inclinación actuando sobre los motores de elevación contrarios.

Asumiendo que se ha establecido el modo ESTABILIZACION en el sensor de inclinación, escriba el código de la rutina de servicio de la interrupción **Int_2** asociada con el mismo. Esta rutina debe compensar la inclinación del aparato actuando inversamente sobre los motores de elevación. Si la inclinación es a la derecha (ángulo

positivo) hay que aumentar la potencia del motor de elevación derecho en una cantidad igual a ese ángulo y disminuir la potencia del izquierdo en la misma cantidad. Para la inclinación a la izquierda se actúa en sentido contrario al anterior, pero el código es el mismo pues en este caso el ángulo es negativo. En caso de inclinación cero hay que establecer la potencia de ambos motores al nivel de sustentación (100).

```
Int_2:  la $t0, 0xFFFF0020    # DB de Sensor inclinación
        lb $t1, 4($t0)         # Lee inclinación en $t1 y cancela
        la $t0, 0xFFFF0000    # DB Control de motores
        bnez $t1, compensar
        li $t2, 100
        sb $t2, 3($t0)        # Potencia 100 a los dos motores de elevación
        sb $t2, 2($t0)
        b retexec
compensar:
        li $t2, 100           # o también lw $t2, Potencia_Elevacion_Nominal
        add $t2, $t2, $t1     # Suma 100 + ángulo inclinación
        sb $t2, 3($t0)        # al motor derecho
        li $t2, 100           # o también lw $t2, Potencia_Elevacion_Nominal
        sub $t2, $t2, $t1     # Resta 100 – ángulo inclinación
        sb $t2, 2($t0)        # al motor izquierdo




       b retexc
```