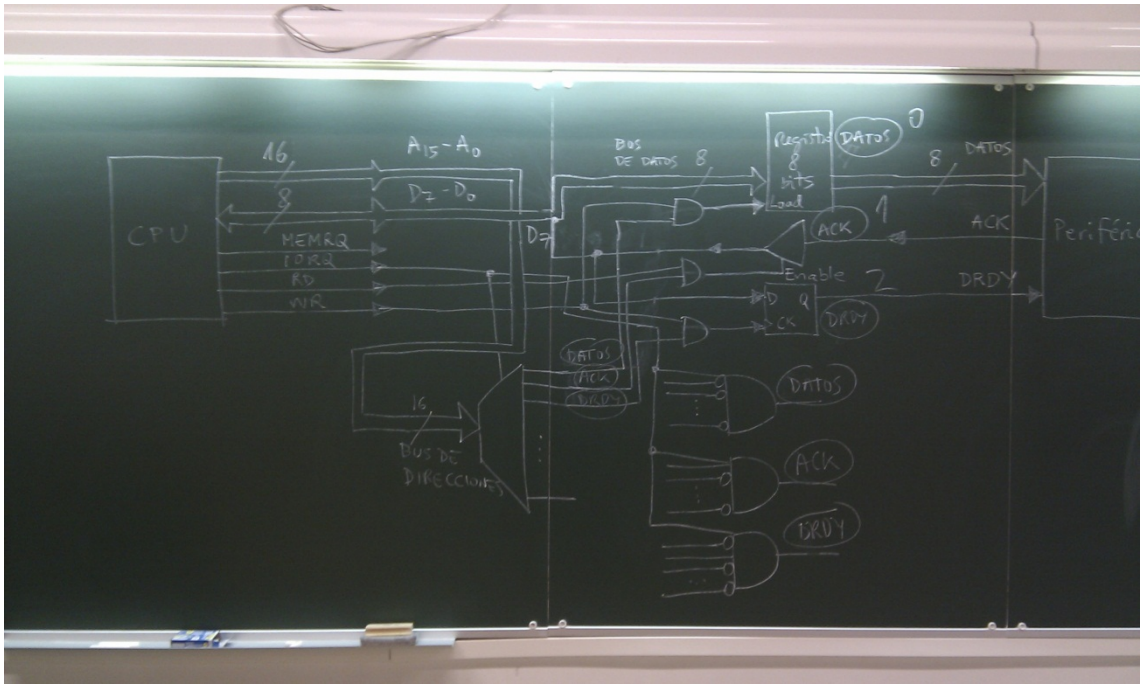
Problema E/S 1

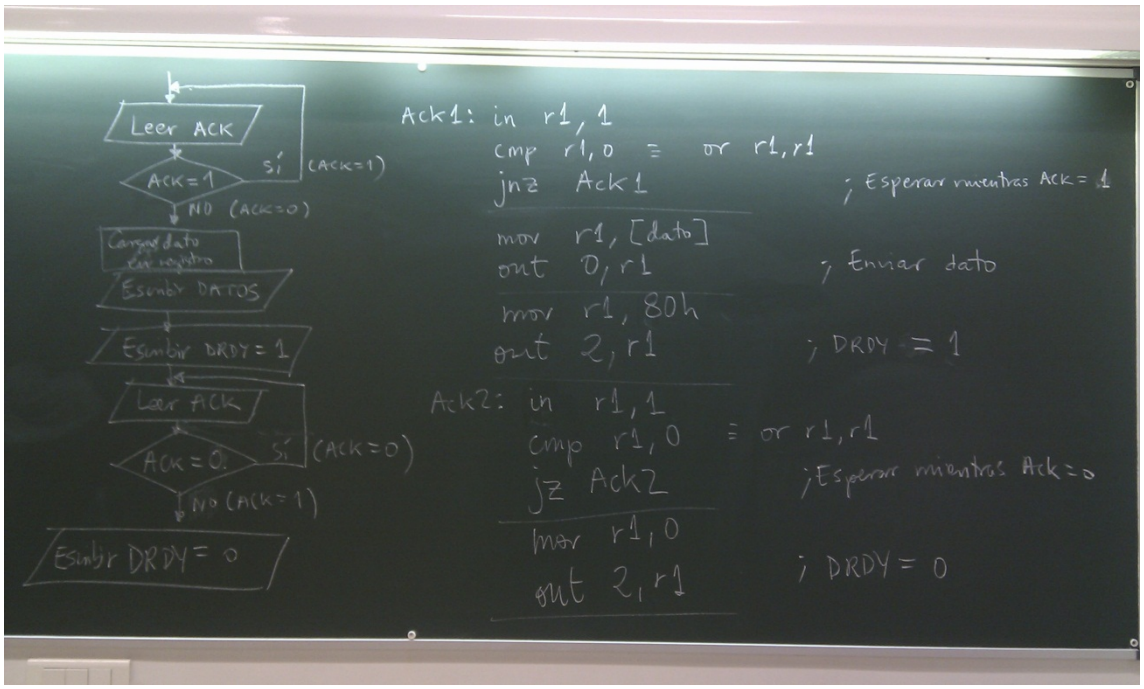
Versión 1

a)

Se inventan los puertos 0,1,2 (decodificador o 3 puertas AND mejor) como puertos de datos (escritura), estado (lectura ACK, lectura directa de la patilla) y "control" (escritura de DRDY). Habría que aclarar temporización CPU WR-addr-data y carga biestable DRDY (y registro DATOS) (y retraso AND) porque se usa AND(addr2,WR) como reloj DRDY, AND(addr0,WR) para DATOS ¿D7/D0-7 se escriben a tiempo?



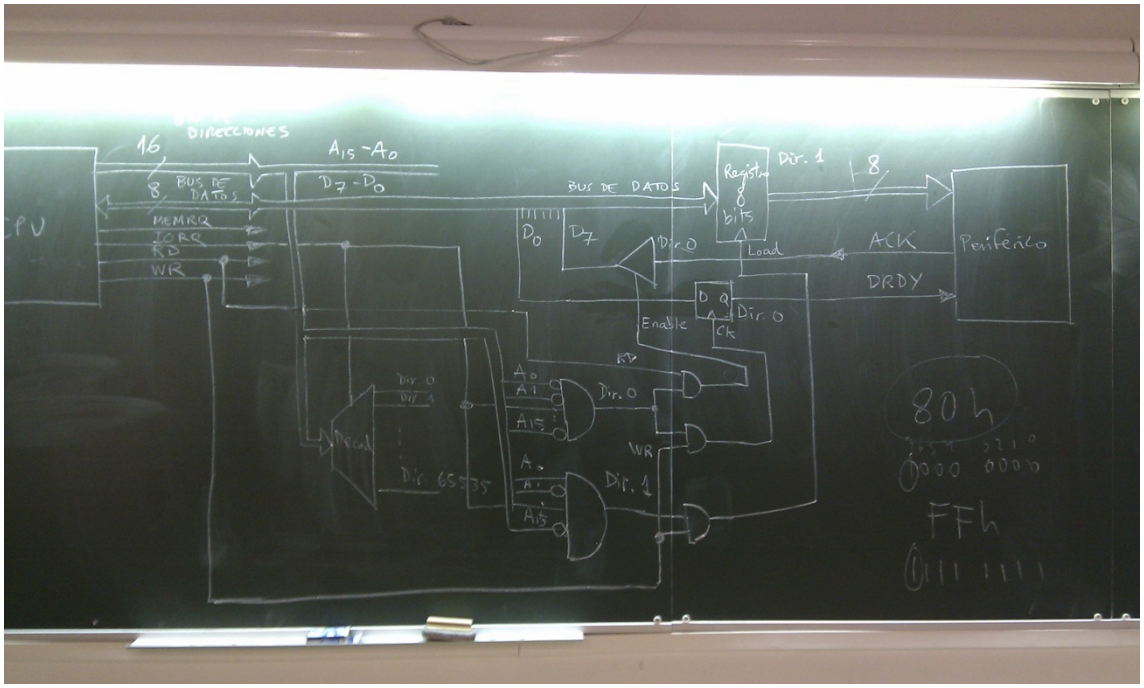
b)



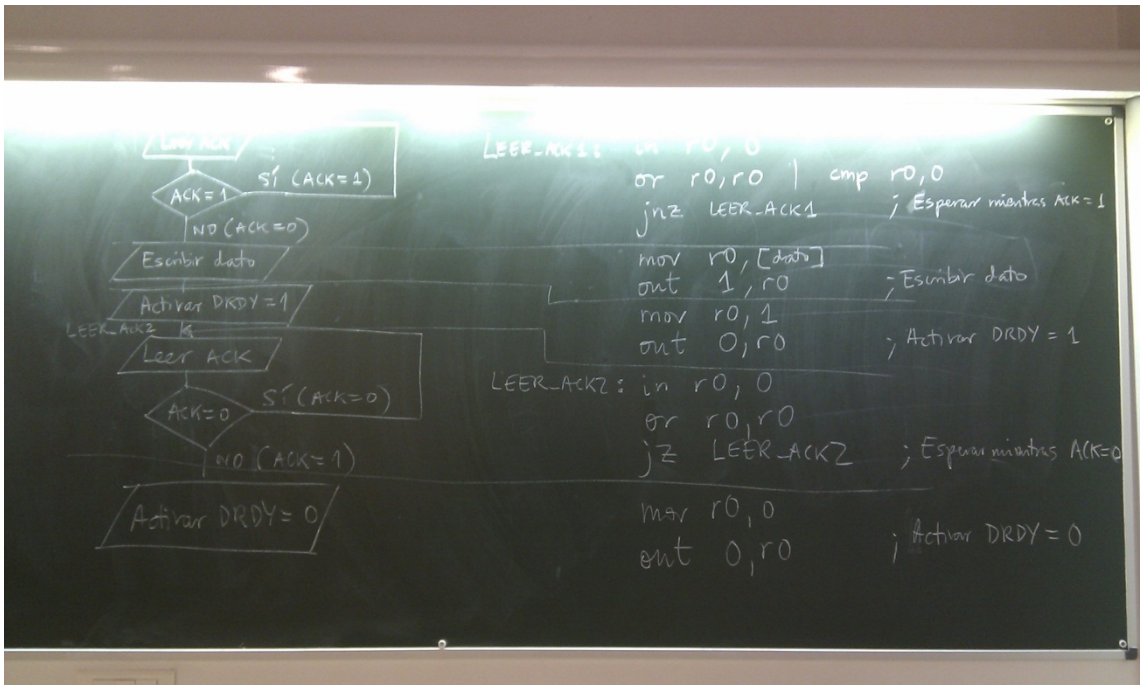
Versión 2

a)

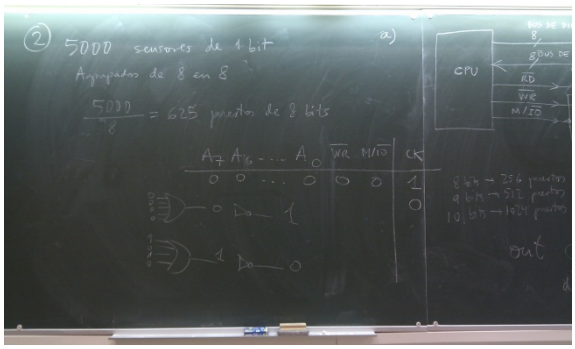
Esta vez se juntan lectura ACK y escritura DRDY en el puerto 0, y escritura DATOS en puerto 1. ACK sigue siendo lectura directa de pin en D7, DRDY escritura en biestable D, pero esta vez conectado al bit D0 (notar diferencias en el código). (Similar preocupación temporización CPU para escribir DRDY/DATOS)



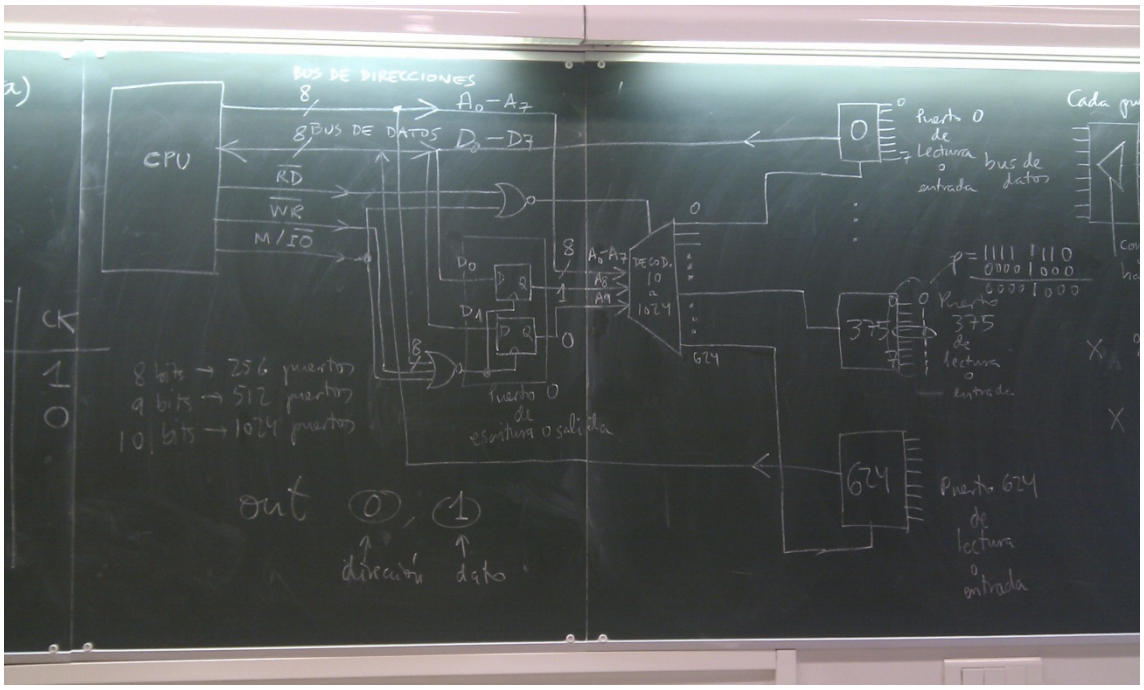
b)

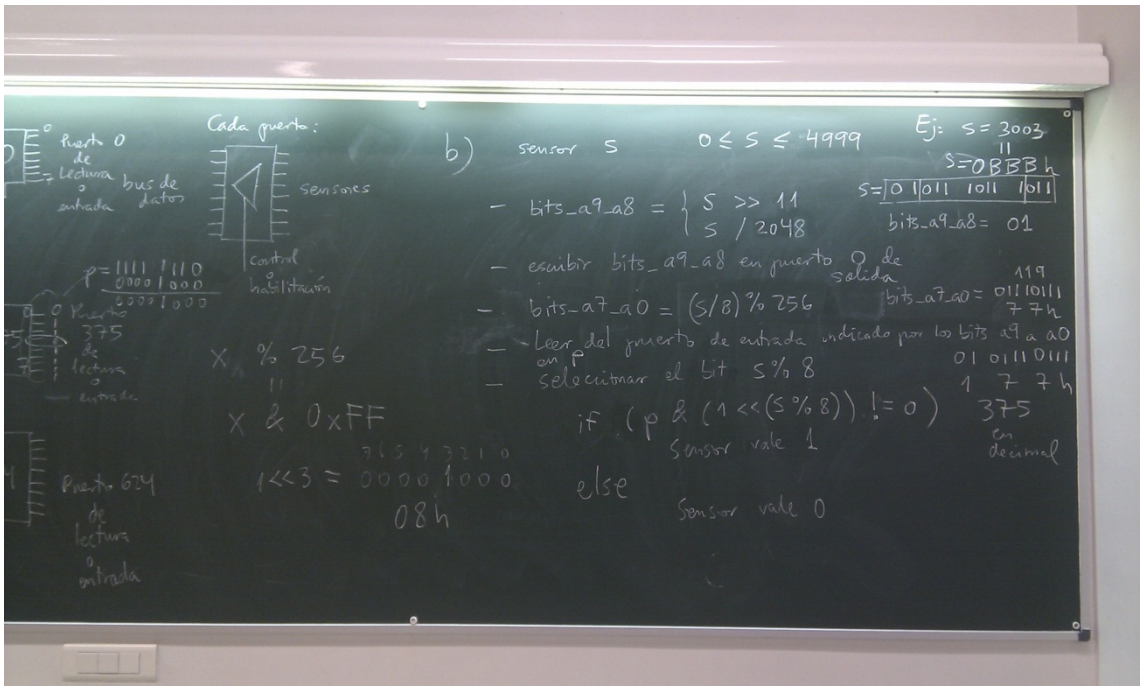


Problema E/S 2

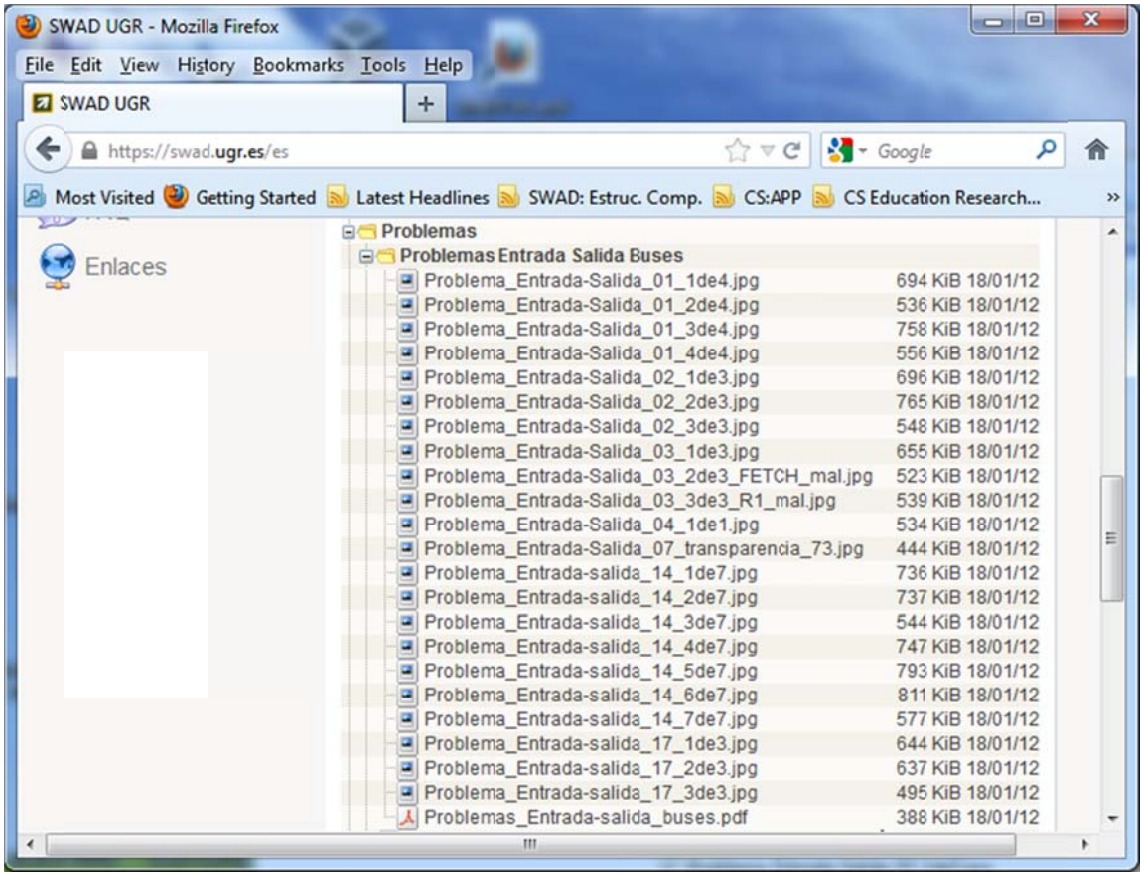


hacen falta 625 puertos y sólo hay 8bits addr, 625 drv.8b (5000bits) puertos entrada 0..624 (drivers tri-estado 8bits, lectura sensor en vivo), se inventa puerto 0 salida solapado con entrada para mantener 2 bits MSB (escritos con D0,D1), puerto 0 se carga con WR IO addr0 (preocup.), notar sensor puede cambiar durante lectura (lectura en vivo, sin buffer)



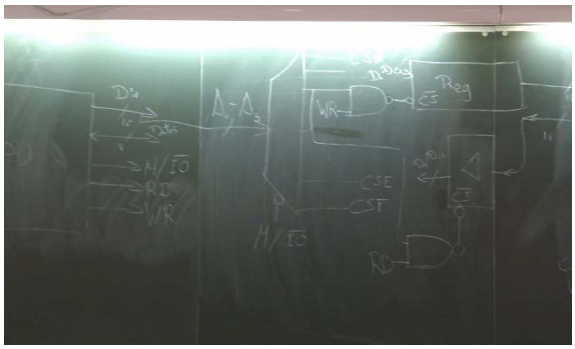


Lista de imágenes

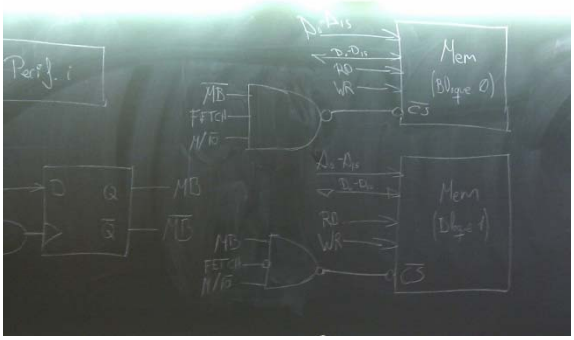


Problema E/S 3

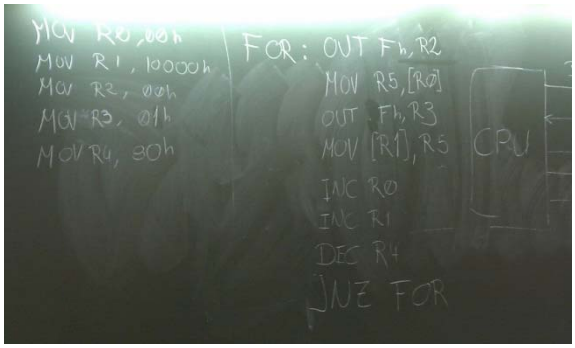
a)



b-c)



d)



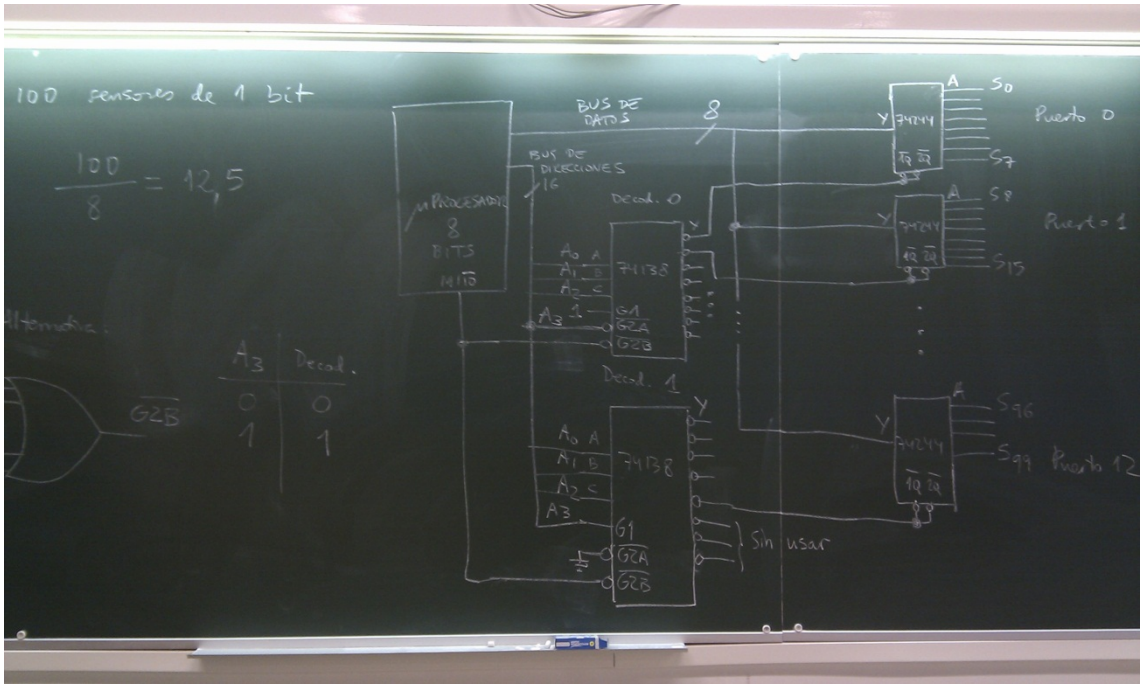
No caben 128KB, hace falta inventarse puerto salida (MB) para seleccionar banco

El espacio de E/S es separado y más reducido, así que no hay problema con mapa memoria. Solapar puerto no es posible (E/S) ni necesario

Copiar del banco 0 al banco 1: en el fuente se ve R1 mal pensado (usar [R0] como src & dst), se deduce que MB es puerto salida Fh, bit D conectado a D0 y R4 es tamaño bloque a copiar

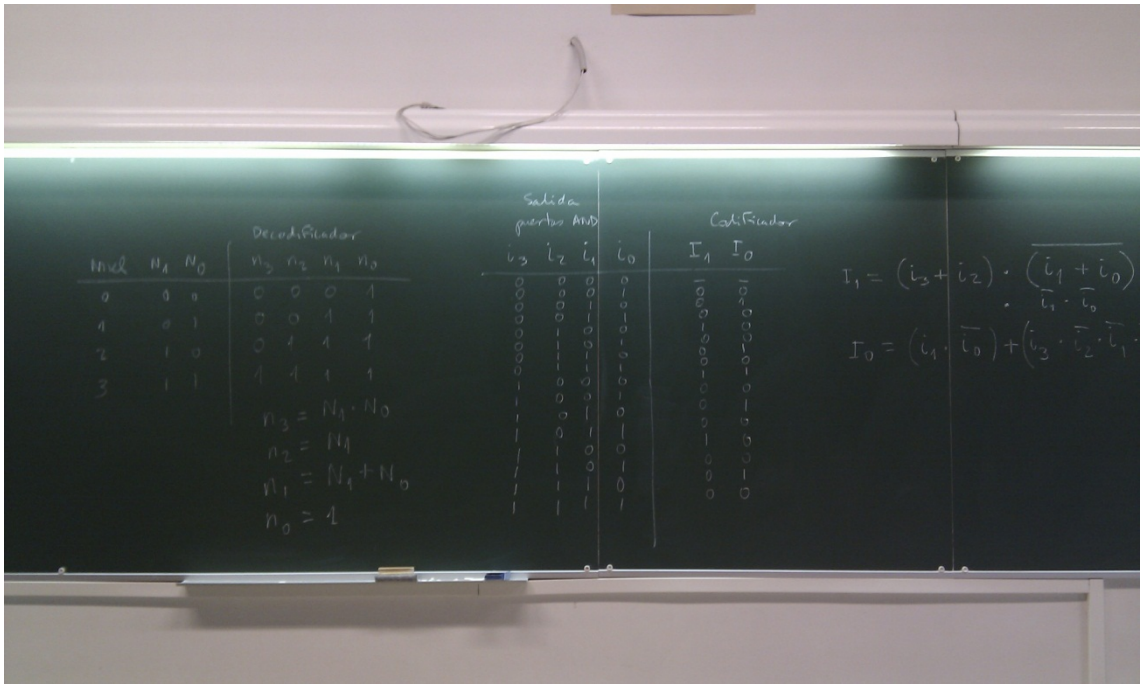
Problema E/S 4

Hacen falta 12.5→13 puertos de entrada. Bus direcciones 16bits como en las CPUs 8bits “tradicionales”. Lectura de alarmas “en vivo”. Decodificación incompleta (bits A0-3). Alternativa completa se ve parte.



Problema E/S 7

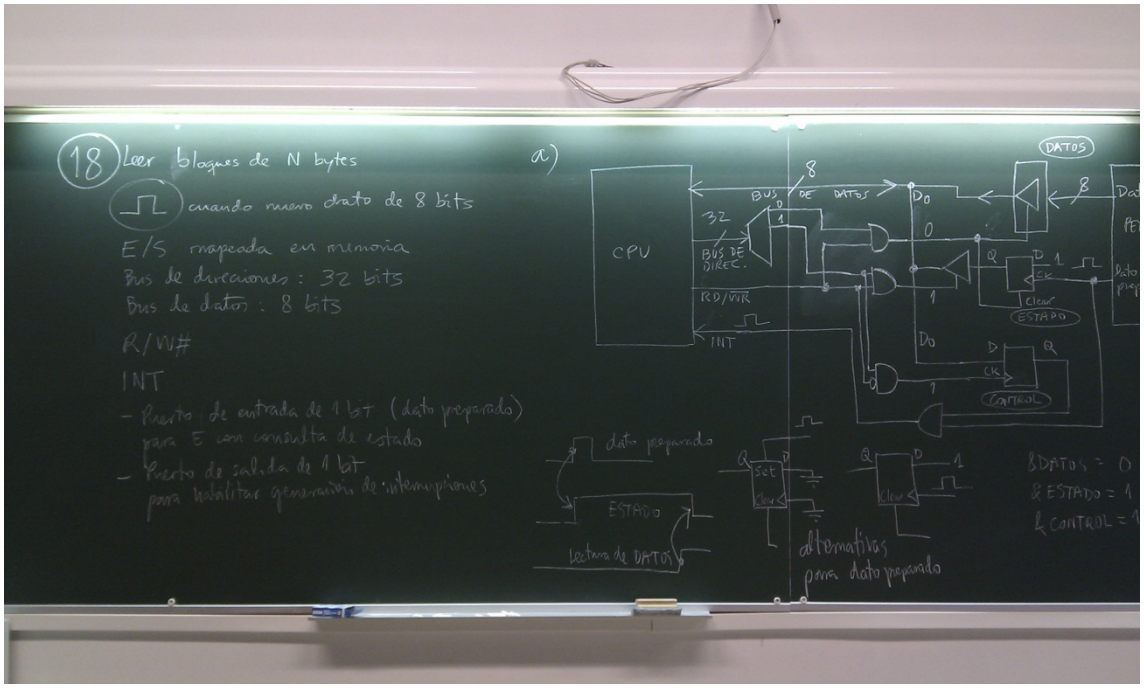
Completar con la transparencia 73 del tema E/S. Puerto de máscara para inhibición selectiva, puerto de nivel y decodificador para inhibición por nivel, bit BGII (CPU) para inhibición total. En la pizarra se detalla el decodificador de nivel, que cuando se fija a un nivel, deshabilita niveles superiores. Nivel max es INT0. El codificador indica la petición de mayor nivel pendiente (max INT0, indiferencia cuando nadie pide)



Problema E/S 14

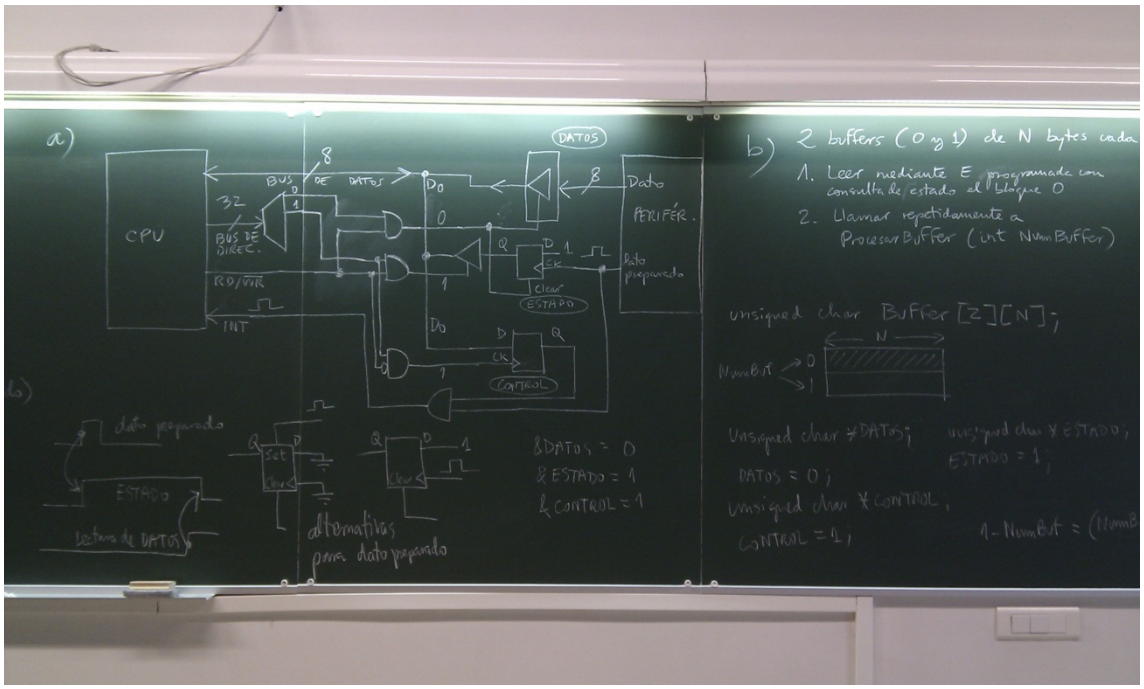
a)

Dir 0 entrada (DAT), habilita buff.3-state. Dir 1 entrada (STAT), habilita biest.D memoriza pulso RDY. Notar que leer dato (DAT) limpia bit RDY. Dir 1 salida (CTRL) carga biest.D memoriza bit D0 (IRQEN). Cuando se activa, el pulso RDY va directo a INT CPU. (Similar preocupación sobre carga IEN)



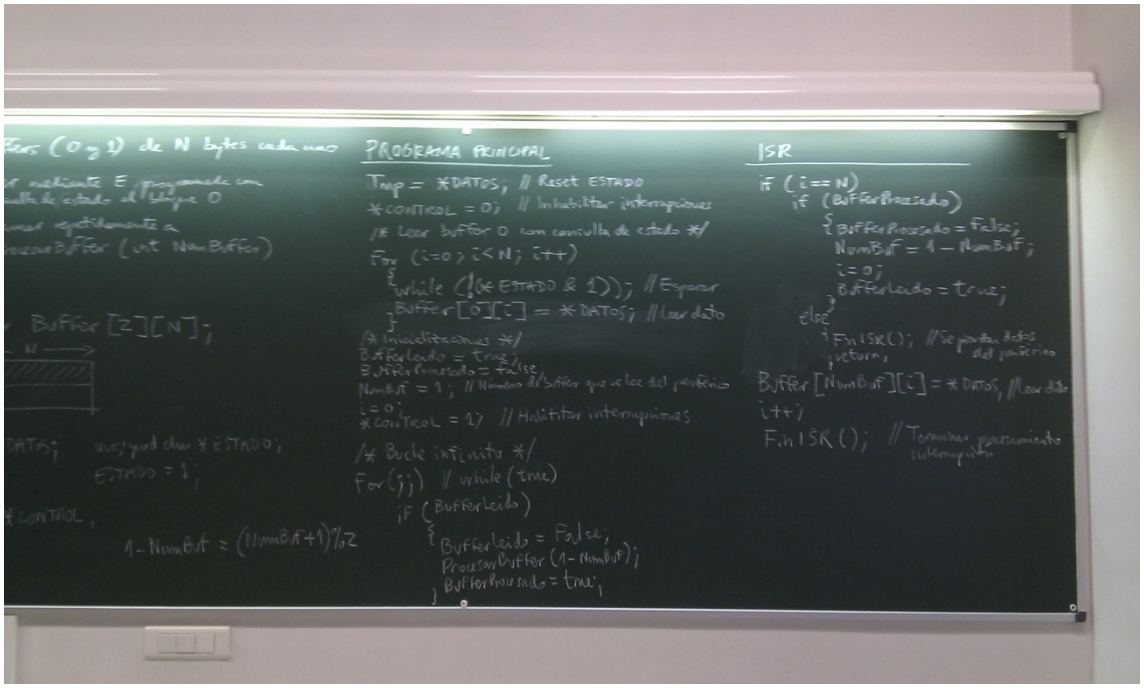
b)

planteamiento: doble buffer[2][N], puntero DATOS, CONTROL, ESTADO (hmf, direcciones fijas memoria), índice NumBuf 0-1



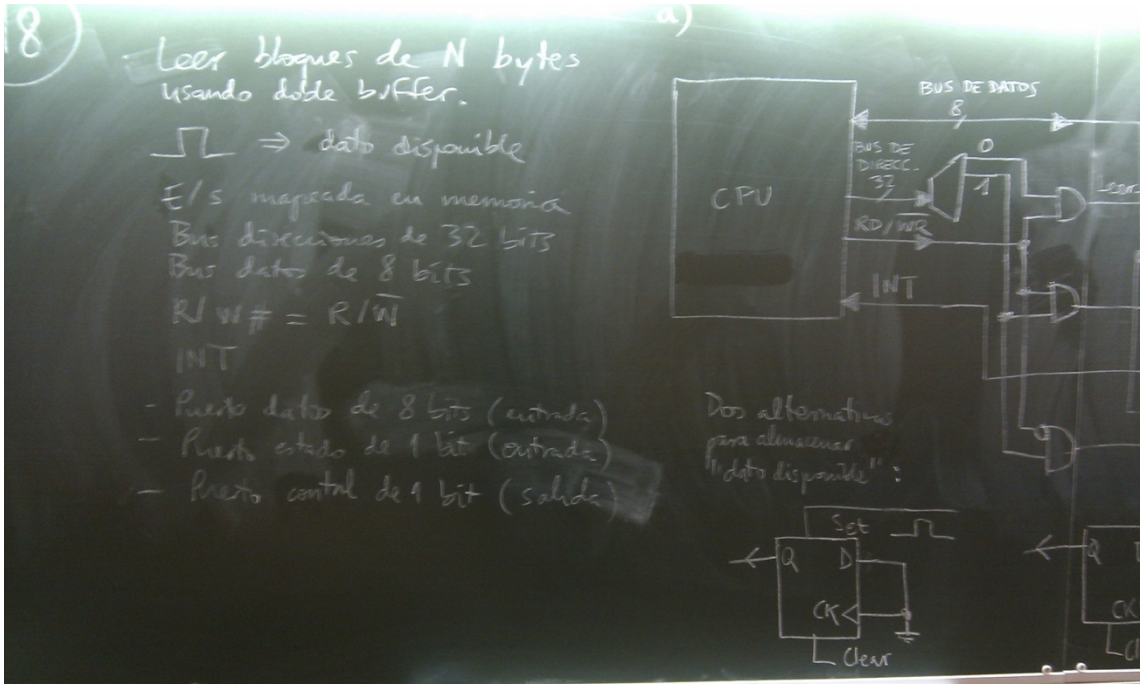
Programa principal: limpia RDY, IEN, lee buffer[0][i] con espera ocupada, prepara interrupciones (BufLeido=1, NumBuf=1, activa IEN), en cuanto hay un buffer(Leido) procesar el que no se está leyendo y marcar que está libre el Buf(Procesado)

ISR: en general guardar DAT en Buffer[NumBuf][i++], hasta N-1 (i++==N, último dato en buffer) en cuyo caso hay que cambiar NumBuf=1-NumBuf, volver a indexar desde i=0, marcar 1 leido sin procesar. Tener 1 flag (BufLeido) permite no procesar hasta buffer entero. Tener contador (BufLeidos=0,1,2) permitiría a la ISR no almacenar hasta que hubiera espacio, si el procesamiento fuera más lento que la lectura.

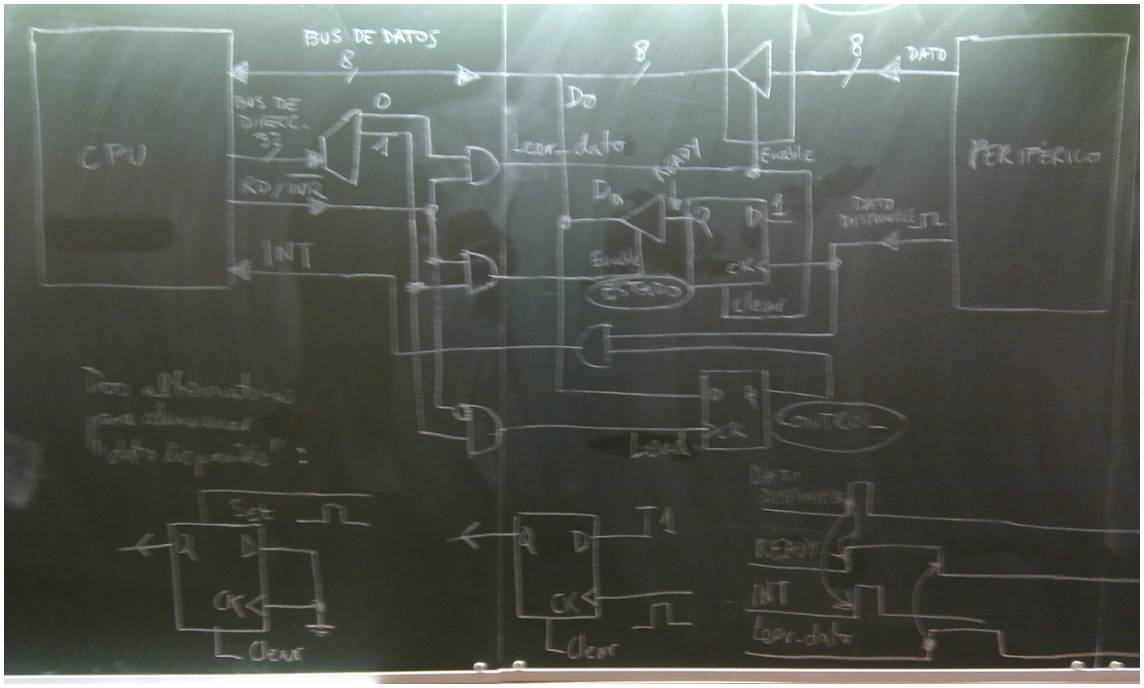


Versión 2

a)

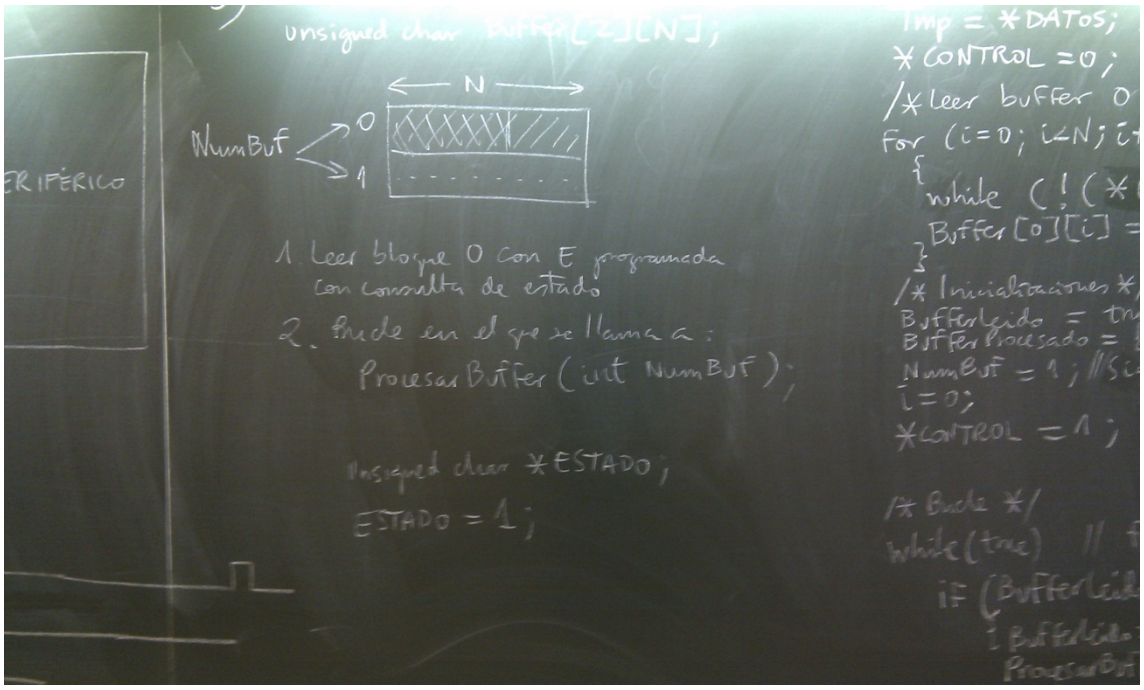


b)

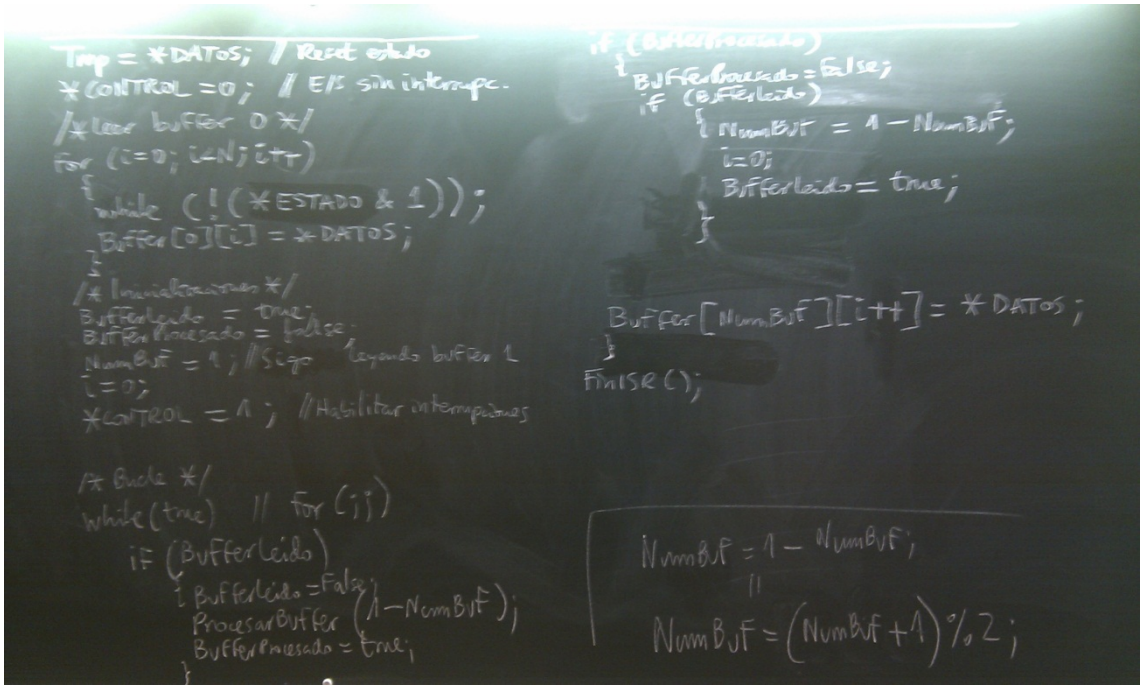


c)

planteamiento



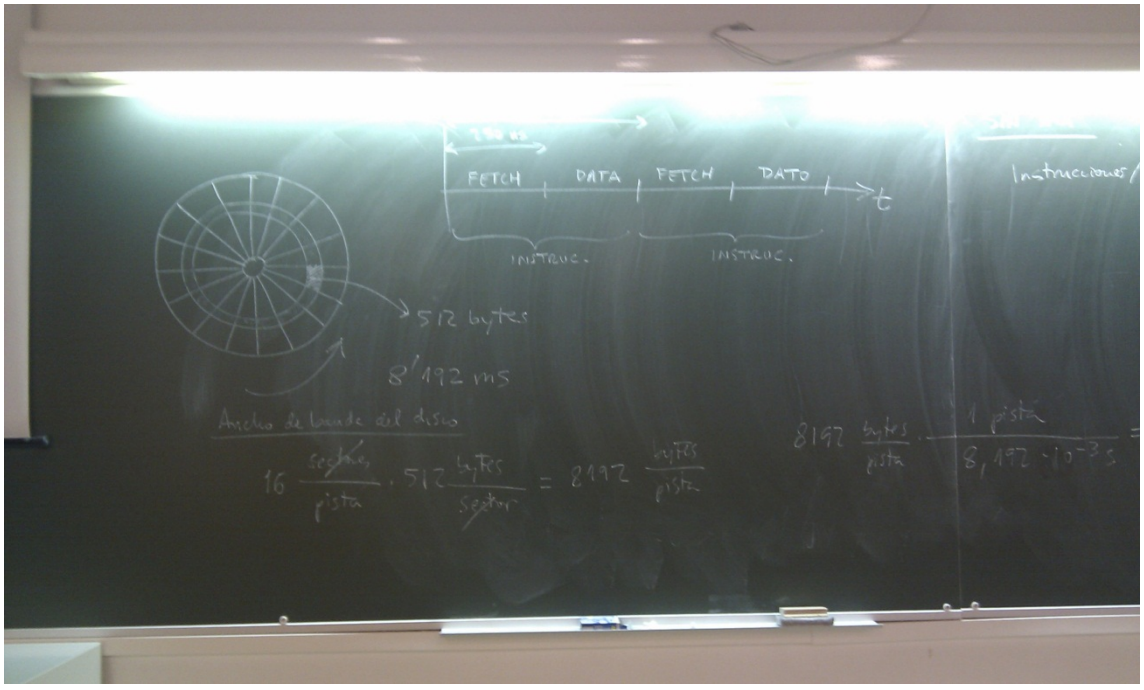
Main - ISR



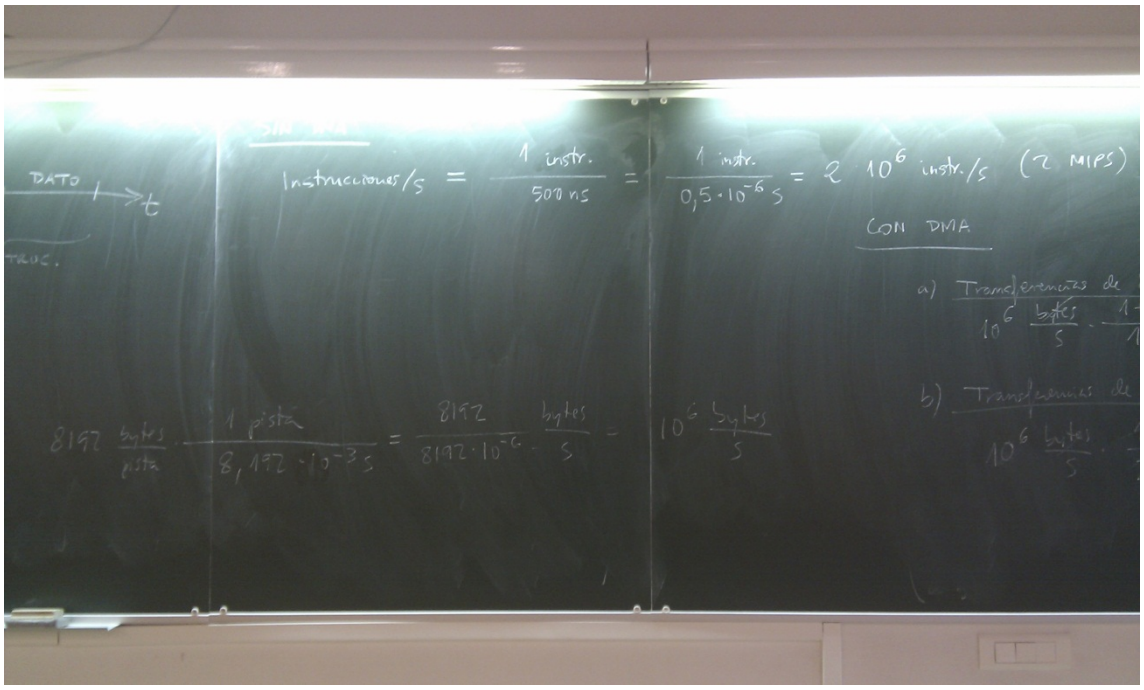
Problema E/S 17

Planteamiento:

Instrucciones 2 ciclos 250ns, disco 8,192ms/rotación, 16 sect/pista → 8KB/pista, 8Kus/rot→1E6 B/s



1instr/500ns→2MIPS



Con robo de ciclo durante 1s se deben gastar 1E6 o 0.5E6 ciclos – la CPU gastaría 4E6 en 1s