Solución examen junio 2020

Estructura de Computadores (Universitat Oberta de Catalunya)

Espacio grapa

Examen 2019/20-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	20/06/2020	15:30

Ficha técnica del examen

- Comprueba que el código y el nombre de la asignatura corresponden a la asignatura matriculada.
- Tiempo total: **2 horas** Valor de cada pregunta: **Se indica en el enunciado**
- En el caso de que los estudiantes no puedan consultar algún material durante el examen, ¿cuáles son?: **NINGUNO**
- Se puede utilitzar calculadora? **NO** De que tipo? **NINGUNO**
- En el caso de que haya preguntas tipo test: ¿descuentan las respuestas erróneas? **NO** ¿Cuánto?

- Indicaciones específicas para la realización de este examen

Examen 2019/20-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	20/06/2020	15:30

Enunciados

No se puede utilizar calculadora. Hay que saber interpretar un valor en binario, decimal o hexadecimal para realizar la operación que se pida. Y el resultado se tiene que expresar en el formato correspondiente.

Valoración de las preguntas del examen

Pregunta 1 (20%)

Pregunta sobre la práctica.
Hay que completar las instrucciones marcadas o añadir el código ensamblador que se pide.
Los puntos suspensivos indican que hay más código, pero no se tiene que completar.
NOTA: Si el código propuesto en cada pregunta no se corresponde con la forma en que vosotros plantearíais la respuesta, podéis rescribir el código o parte del código según vuestro planteamiento.

1.1 : 10%
1.2 : 10%

Pregunta 2 (35%)

2.1 : 10%
2.2 : 15%
2.3 : 10%

Pregunta 3 (35%)

3.1 : 15%
3.1.1 : 10%
3.1.2 : 5%

3.2: 20%
3.2.1 : 10%
3.2.2 : 5%
3.2.3 : 5%

Pregunta 4 (10%)

4.1 : 5%
4.2 : 5%

Examen 2019/20-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	20/06/2020	15:30

Pregunta 1

1.1 Práctica – 1a Parte

Escribir un fragmento de código ensamblador de la subrutina copyMatrixP1 que hace el bucle interior para recorrer las columnas de la matriz mientras se copia la matriz m sobre mRotated. (No se tiene que escribir el código de toda la subrutina)

```
;;;;;
; Copiar los valores de la matriz (mRotated) a la matriz (m).
; La matriz (mRotated) no se tiene que modificar, los cambios se tienen que hacer en la matriz (m).
; Para recorrer la matriz en ensamblador el índice va de 0 (posición [0][0])a 30 (posición [3][3])
; con incrementos de 2 porque los datos son de tipo short(WORD) 2 bytes.
; No se muestra la matriz.
; Variables globales utilizadas:
; m        : Matriz 4x4 donde hay los números del tablero de juego.
; mRotated : Matriz 4x4 para hacer la rotación.
;;;;;
copyMatrixP1:
   push rbp
   mov  rbp, rsp
   ...
   mov rax, 0       ;index [i][j] matriz m y mRotated
   mov r8, 0                      ;i=0
   copyMatrixP1_fori:
   cmp r8, DimMatrix              ;i<DimMatrix
   jge copyMatrixP1_endfori
                                  ; for (j=0; j<DimMatrix; j++) {
                                  ;    m[i][j] = mRotated[i][j];
                                  ; }

      mov r9, 0                           ;    j=0
      copyMatrixP1_forj:
      cmp r9, DimMatrix                   ;j<DimMatrix
      jge copyMatrixP1_endforj

      mov bx, WORD[mRotated+rax]          ;mRotated[i][j]
      mov WORD[m+rax], bx                 ;m[i][j] = mRotated[i][j];

      add rax, 2                          ;index+2
      inc r9                              ;j++


      jmp  copyMatrixP1_forj
      copyMatrixP1_endforj:
      inc r8                         ;i++
      jmp copyMatrixP1_fori
   copyMatrixP1_endfori:
   ...
   mov rsp, rbp
   pop rbp
   ret
```

1.2 Práctica – 2a parte

Examen 2019/20-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	20/06/2020	15:30

Completar el código de la subrutina updateBoardP2. (Sólo completar los espacios marcados, no se pueden añadir o modificar otras instrucciones).

```
;;;;;
; Convertir el valor recibido como parámetro (edx) de tipo int (DWORD) de 6 dígitos (num<=999999)
; a los caracteres ASCII que representen su valor.
; Hay que dividir el valor entre 10, de forma iterativa, hasta obtener 6 dígitos.
; En cada iteración, el residuo de la división que es un valor entre (0-9) indica el valor del
; dígito que tenemos que convertir a ASCII ('0'-'9') sumando '0' (48 decimal) para poderlo mostrar.
; Cuando el cociente sea 0 mostraremos espacios en la parte no significativa.
; Por ejemplo, si number=103 mostraremos "   103" y no "000103".
; Se tienen que mostrar los dígitos (carácter ASCII) desde la posición indicada por la fila (edi) y
; la columna (esi) recibidos como parámetro, posición de les unidades, hacia la izquierda.
; Como el primer dígito que obtenemos son las unidades, después las decenas, ..., para mostrar el
; valor se tiene que desplazar el cursor una posición a la izquierda en cada iteración.
; Para posicionar el cursor llamar a la subrutina gotoxyP2 y para mostrar los caracteres a
; la subrutina printchP2 implementando correctamente el paso de parámetros.
; Variables globales utilizadas: Ninguna.
; Parámetros de entrada: rdi (edi): Fila donde lo queremos mostrar en pantalla.
;                        rsi (esi): Columna donde lo queremos mostrar en pantalla.
;                        rdx (edx): Valor que queremos mostrar en pantalla.
; Parámetros de salida : Ninguno.
;;;;;
showNumberP2:

;;;;;
; Actualizar el contenido del Tablero de Juego con los datos de la matriz (m) de 4x4 valores de
; tipo short (WORD) y los puntos del marcador (score) que se han hecho.
; Se tiene que recorrer toda la matriz (m), y para cada elemento de la matriz posicionar el cursor
; en pantalla y mostrar el número de esa posición de la matriz.
; Después, mostrar el marcador que recibimos como parámetro (edi) en la parte inferior del tablero,
; fila 18, columna 26 llamando a la subrutina showNumberP2.
; Finalmente posicionar el cursor en la fila 18, columna 28 llamando a la subrutina gotoxyP2.
; Variables globales utilizadas: m : matriz 4x4 donde hay los números del tablero de Juego.
; Parámetros de entrada: rdi (edi): puntos acumulados hasta el momento.
; Parámetros de salida : Ninguno.
;;;;;
updateBoardP2:
   push rbp
   mov  rbp, rsp
   ...
   mov r8d, edi     ;src;
   mov eax, 0       ;i
   mov ebx, 0       ;j
   mov edi, 0       ;rScreen
   mov esi, 0       ;cScreen
   mov rcx, 0       ;índex per a accedir a la matriu m. (0-15)
                    ;índex=(fila*DimMatrix)+(columna)*4


   ;Iniciem el bucle per a mostrar la matriu.
   mov edi, 10      ;rScreen = 10;
   mov eax, 0       ;i=0;
   updateBoardP2_fori:
   cmp eax, DimMatrix    ;i<DimMatrix;
   jge updateBoardP2_endfori

      mov esi, 17        ;cScreen = 17;



      mov ebx, 0         ;j=0;
      updateBoardP2_forj:
      cmp ebx, DimMatrix ;j<DimMatrix;
      _jge_ updateBoardP2_endforj
         mov   edx, 0
```

Examen 2019/20-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	20/06/2020	15:30

```
        mov    dx, _WORD[m+rcx]_  ;m[i][j];
        call showNumberP2    ;showNumberP2_C(rScreen,cScreen, m[i][j]);

      add esi, 9         ;cScreen = cScreen + 9;
      add rcx, _2_          ;index
      inc ebx            ;j++.

    jmp  updateBoardP2_forj

    updateBoardP2_endforj:

    add edi, 2         ;rScreen = rScreen + 2;
    inc eax            ;i++.

    jmp updateBoardP2_fori

updateBoardP2_endfori:

mov   _edi_, 18
mov   _esi_, 26
mov  edx, r8d          ;src;
call showNumberP2      ;showNumberP2_C(18, 26, scr);
mov  esi, 28
_call_ gotoxyP2        ;gotoxyP1_C(18,28);
...
mov rsp, rbp
pop rbp
ret
```

Examen 2019/20-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	20/06/2020	15:30

Pregunta 2

2.1

El estado inicial del computador CISCA justo antes de empezar la ejecución de cada fragmento de código (a cada apartado) es el siguiente:

R0 = 00000000h R1 = 00000A10h R2 = 00000B20h R3 = 00000C30h R4 = 00000D40h	M(00000A10h) = F0000000h M(00000B20h) = 80000000h M(00000C30h) = 0000FFFFh M(000000F0h) = 00000001h M(00000D40h) = 00000010h	Z = 0, C = 0, S = 0, V = 0

¿Cuál será el estado del computador después de ejecutar cada fragmento de código? (sólo modificaciones, excluyendo el PC).

a)

```
        MOV R0, [R1]
        ADD R0, [R2]
```

R0= F00000000h
R0= 700000000h

Z = 0 , S = 0 , C = 1 , V = 1

b)

```
        CMP [R1], R2
        JG FIN
        SAR R3, [R4]
FIN:
```

R3= 00000000h

Z = 1 , S = 0 , C = 0 , V = 0

Examen 2019/20-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	20/06/2020	15:30

2.2

Dado el siguiente código de alto nivel:

Si (A[i+1]==A[i]) A[i]= A[i+1] * 2;

A es un vector de 8 elementos de 4 bytes cada uno. Se propone la siguiente traducción a CISCA donde hemos dejado 6 espacios para llenar.

```
      MOV R0, [i]
      MUL R0, 4
      ADD R0, 4
      MOV R2, [A+R0]
      DEC R0, 4
      CMP R2,[A+R0]
     JNE END
      MOV [A+R0], R2
      SAL [A+R0], 1
END:
```

Examen 2019/20-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	20/06/2020	15:30

2.3

Dado el siguiente fragmento de código de un programa en lenguaje ensamblador de CISCA:

```
        ADD R0, [R2+100h]
        JE END
        MUL R0, [R4]
        SAL R10, [A]
END:
```

Traducidlo a lenguaje máquina y expresadlo en la siguiente tabla. Suponed que la primera instrucción del código se ensambla a partir de la dirección 00006880h (que es el valor del PC antes de empezar la ejecución del fragmento de código). Suponed que la dirección simbólica A vale 00004000h. En la siguiente tabla usad una fila para codificar cada instrucción. Si suponemos que la instrucción comienza en la dirección @, el valor Bk de cada uno de los bytes de la instrucción con direcciones @+k para k=0, 1,… se debe indicar en la tabla en hexadecimal en la columna correspondiente (recordad que los campos que codifican un desplazamiento en 2 bytes o un inmediato o una dirección en 4 bytes lo hacen en formato little endian, esto hay que tenerlo en cuenta escribiendo los bytes de menor peso, de dirección más pequeña, a la izquierda y los de mayor peso, dirección mayor, a la derecha). Completad también la columna @ que indica para cada fila la dirección de memoria del byte B0 de la instrucción que se codifica en esa fila de la tabla.

A continuación os damos como ayuda las tablas de códigos:

Tabla de códigos de instrucción

B0	Instrucción
35h	SAL
22h	MUL
20h	ADD
43h	JE

Tabla de modos de direccionamiento (Bk<7..4>)

Campo modo Bk<7..4>	Modo
0h	Inmediato
1h	Registro
2h	Memoria
3h	Indirecto
4h	Relativo
5h	Indexado
6h	Relativo a PC

Tabla de modos de direccionamiento (Bk<3..0>)

Campo modo Bk<3..0>	Significado
Nº registro	Si el modo tiene que especificar un registro
0	No se especifica registro.

		Bk para k=0..10										
@	**Ensamblador**	0	1	2	3	4	5	6	7	8	9	10
00006880h	ADD R0,[R2+100h]	20	10	42	00	01						
00006885h	JE END	43	60	0A	00							
00006889h	MUL R0, [R4]	22	10	34								
0000688Ch	SAL R10, [A]	35	1A	20	00	40	00	00				

Examen 2019/20-2

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	20/06/2020	15:30

00006893h												

Examen 2019/20-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	20/06/2020	15:30

Pregunta 3

3.1. Memoria cache

Tenemos un sistema de memoria en el que todos los accesos se realizan a palabra (no nos importa cuál es el tamaño de la palabra). Supondremos que el espacio de direcciones de memoria se descompone en bloques de 8 palabras. Cada bloque empieza en una dirección múltiplo de 8. Así, el bloque 0 contiene las direcciones 0, 1, 2, 3, 4, 5, 6, 7, el bloque 1, las direcciones 8, 9, 10, 11, 12, 13, 14, 15, y el bloque N las direcciones 8*N, 8*N+1, 8*N+2, 8*N+3, 8*N+4, 8*N+5, 8*N+6, 8*N+7.

Suponemos que el sistema también dispone de una memoria cache de 4 líneas (donde cada línea tiene el tamaño de un bloque, es decir, 8 palabras). Estas líneas se identifican como líneas 0, 1, 2 y 3. Cuando se hace una referencia a una dirección de memoria principal, si esta dirección no se encuentra en la memoria cache, se trae todo el bloque correspondiente desde la memoria principal a una línea de la memoria cache (así si hacemos referencia a la dirección 2 de memoria principal traeremos el bloque formado por las palabras 0, 1, 2, 3, 4, 5, 6, 7).

Suponemos que el sistema utiliza una política de emplazamiento completamente asociativa, de manera que cualquier bloque de la memoria principal se puede llevar a cualquier bloque de la memoria cache. Si encontramos que la memoria cache ya está llena, se utiliza un **algoritmo de reemplazamiento LRU.**

La ejecución de un programa genera la siguiente lista de lecturas a memoria:

12, 13, 25, 26, 17, 8, 22, 3, 24, 62, 5, 63, 64, 17, 18, 19, 57, 58, 20, 25

3.1.1.

La siguiente tabla muestra el estado inicial de la cache, que contiene las primeras 32 palabras de la memoria (organizadas en 4 bloques).

Completar la tabla para mostrar la evolución de la cache durante la ejecución del programa. Para cada acceso se debe rellenar una columna indicando si se trata de un acierto o un fallo.

Si es un acierto escribiremos E en la línea correspondiente delante de las direcciones del bloque, si es un fallo escribiremos F i se indicará el nuevo bloque que es trae a la memoria cache en la línea que le corresponda, expresando de la forma b (a_0 - a_7) donde b: número de bloque, y (a_0 - a_7) son las direcciones del bloque, donde a_0 es la primera dirección del bloque y a_7 es la octava (última) dirección del bloc.

Línea	Estado Inicial	12	13	25	26	17
0	0 (0 - 7)	0 (0 - 7)	0 (0 - 7)	0 (0 - 7)	0 (0 - 7)	0 (0 - 7)
1	1 (8 - 15)	E 1 (8 - 15)	E 1 (8 - 15)	1 (8 - 15)	1 (8 - 15)	1 (8 - 15)
2	2 (16 - 23)	2 (16 - 23)	2 (16 - 23)	2 (16 - 23)	2 (16 - 23)	E 2 (16 - 23)
3	3 (24 - 31)	3 (24 - 31)	3 (24 - 31)	E 3 (24 - 31)	E 3 (24 - 31)	3 (24 - 31)

Examen 2019/20-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	20/06/2020	15:30

Línea	8	22	3	24	62	5
0	0 (0 - 7)	0 (0 - 7)	E 0 (0 - 7)	0 (0 - 7)	0 (0 - 7)	E 0 (0 - 7)
1	E 1 (8 - 15)	1 (8 - 15)	1 (8 - 15)	1 (8 - 15)	F 7 (56 - 63)	7 (56 - 63)
2	2 (16 - 23)	E 2 (16 - 23)	2 (16 - 23)	2 (16 - 23)	2 (16 - 23)	2 (16 - 23)
3	3 (24 - 31)	3 (24 - 31)	3 (24 - 31)	E 3 (24 - 31)	3 (24 - 31)	3 (24 - 31)

Línea	63	64	17	18	19	57
0	0 (0 - 7)	0 (0 - 7)	0 (0 - 7)	0 (0 - 7)	0 (0 - 7)	0 (0 - 7)
1	E 7 (56 - 63)	7 (56 - 63)	7 (56 - 63)	7 (56 - 63)	7 (56 - 63)	E 7 (56 - 63)
2	2 (16 - 23)	F 8 (64 - 71)	8 (64 - 71)	8 (64 - 71)	8 (64 - 71)	8 (64 - 71)
3	3 (24 - 31)	3 (24 - 31)	F 2 (16 - 23)	E 2 (16 - 23)	E 2 (16 - 23)	2 (16 - 23)

Línea	58	20	25			
0	0 (0 - 7)	0 (0 - 7)	F 3 (24 – 31)			
1	E 7 (56 - 63)	7 (56 - 63)	7 (56 - 63)			
2	8 (64 - 71)	8 (64 - 71)	8 (64 - 71)			
3	2 (16 - 23)	E 2 (16 - 23)	2 (16 - 23)			

3.1.2 a)

¿Cuál es la tasa de aciertos (T_a)?

T_a = 16 aciertos / 20 accesos = 0,8

3.1.2 b)

Suponemos que el tiempo de acceso a la memoria cache, o tiempo de acceso en caso de acierto (t_e), es de 5 ns y el tiempo total de acceso en caso de fallo (t_f) es de 40 ns. Considerando la tasa de aciertos obtenida en la pregunta anterior, ¿cuál es el tiempo medio de acceso a memoria (t_m)?

$t_m = T_e \times t_e + (1-T_e) \times t_f$ = 0,8 * 5 ns + 0,2 * 40 ns = 4 ns + 8 ns = 12 ns

Examen 2019/20-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	20/06/2020	15:30

3.2 Sistema d'E/S

Se quiere analizar el rendimiento de la comunicación de datos entre una memoria de un procesador y un puerto USB, que tienen las siguientes características, utilizando E/S por interrupciones:

2 Velocidad de transferencia del dispositivo de E/S ($\mathbf{v}_{transf}$)= 1 MBytes/s = 1000 Kbytes/s
3 Tiempo de latencia media del dispositivo ($t_{latencia}$) = 0
4 Direcciones de los **registros de datos** y **de estado** del controlador de E/S: 0B00h i 0B04h
5 El bit del **registro de estado** que indica que el controlador del puerto de E/S está disponible es el bit 2, o el tercer bit menos significativo (cuando vale 1 indica que está disponible)
6 Procesador con una frecuencia de reloj de 1 GHz, el procesador puede ejecutar 4 instrucciones por ciclo de reloj.
7 Transferencia de **lecture** desde puerto de E/S hacia la memoria
8 Transferencia de N_{datos}=100.000 datos
9 La medida de un dato es m_{dada} = 4 bytes
10 El tiempo para atender la interrupción (t_{rec_int}) es de 2 ciclos de reloj

3.2.1 Completar el siguiente código CISCA que es una rutina de servicio a las interrupciones (RSI) para transferir a través del dispositivo de E/S anterior, mediante la técnica de E/S por interrupciones.

Se utiliza una variable global que se representa con la etiqueta **Addr,** y que al principio del programa contiene la dirección inicial de memoria donde almacenar los datos recibidos.

```
1.   CLI
2.   PUSH R0
3.   PUSH R1
4.   IN   R0, [0B04h]
5.   MOV  R1, [Addr]
6.   MOV  [R1],R0
7.   ADD  R1,4
8.   MOV  [Addr],R1
9.   POP  R1
10.  POP  R0
11.  STI
12.  IRET
```

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	20/06/2020	15:30

3.2.2 ¿Cuánto tiempo dura la transferencia del bloque de datos t_{transf_bloc}?

El tiempo de un ciclo, t_{ciclo} = 1 ns (nanosegundos)

Tiempo para atender la interrupción, t_{rec_int}: 2 cicles * 1 ns = 2 ns

Tiempo de ejecución de una instrucción, t_{instr}: t_{ciclo} / 4 = 0,250 ns

Tiempo de ejecución RSI, t_{rsi}: N_{rsi} * t_{instr} = 12 instr. * 0,250 ns = 3 ns

Tiempo consumido por la CPU en cada interrupción, t_{transf_dada}:

t_{transf_dada} = t_{rec_int} + t_{rsi} = 2 + 3 = 5 ns

Número de interrupciones producidas (número total de datos, N_{datos}): 100000 interrupciones

Tiempo consumido en total por TODAS las interrupciones:

t_{transf_bloc} = t_{transf_dada} * N_{dades} = 5 ns * 100000 interrupciones = 500000 ns = 0,5 ms (milisegundos)

3.2.3 ¿Cuál es el porcentaje de ocupación del procesador? Porcentaje que representa el tiempo de transferencia del bloc t_{transf_bloc} respecto al tiempo de transferencia del bloque por parte del periférico t_{bloc}

t_{bloc} = $t_{latencia}$ + (N_{datos} * t_{dato})
$t_{latencia}$ = 0
N_{datos} = 100000
t_{dato} = m_{dato} / v_{transf} = 4 / 1000 Kbytes/s = 0,004 ms
t_{bloc} = 0 + (100000 * 0,004) ms = 400 ms
% ocupación = (t_{transf_bloc} * 100 / t_{bloc}) = (0,5 * 100) / 400 = 0,125%

Examen 2019/20-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	20/06/2020	15:30

Pregunta 4

4.1

¿Qué diferencia hay en ensamblador entre un salto condicional y otro incondicional? ¿Qué operando codificamos en la instrucción en cada caso? Pon algun ejemplo de ambos tipos.

Los saltos condicionales son aquellos que toman la decisión de salto consultando los bits de estado en función de una condición codificada en la instrucción. Junto con el código de operación se codifica el desplazamiento o número de bits que se han de saltar, hacia adelante o hacia atrás. Por ejemplo JE, JNE, JG son instrucciones de salto condicional.

Los saltos incondicionales no dependen de una condición. Se codifica la dirección real de salto. Por ejemplo JMP.

4.2

a) En la memoria cache, ¿Qué políticas de asignación se definen? Describirlas brevemente.

1) Política de asignación directa: un bloque de la memoria principal sólo puede estar en una única línea de la memoria cache. La memoria cache de asignación directa es la que tiene la tasa de fallos más alta, pero se utiliza mucho porqué es la más barata y fácil de gestionar

2) Política de asignación completamente asociativa: un bloque de la memoria principal puede almacenarse en cualquier línea de la memoria cache. La memoria cache completamente asociativa es la que tiene la tasa de fallos más baja. A pesar de eso, no se suele utilizar porqué es la más cara y compleja de gestionar.

3) Política de asignación asociativa por conjuntos: un bloque de la memoria principal puede almacenarse en un subconjunto de las líneas de la memoria cache, pero dentro del subconjunto puede encontrarse ne cualquier posición. La memoria cache asociativa por conjuntos es una combinación.

b) ¿Cuáles son los pasos básicos para la gestión de una interrupción en un sistema con una única línea de interrupción y un único módulo de E/S?

1.- Petición del módulo de Entrada/Salida
2.- Ciclo de reconocimiento de la interrupción
    2.a.- Reconocimiento de la interrupción
    2.b.- Salvaguarda del estado del procesador
    2.c.- Llamada a la RSI
3.- Ejecución de la rutina de servicio de interrupción
    3.a.-Inicio de la ejecución de la RSI
    3.b.- Intercambio del dato
    3.c Finalización de la ejecución de la RSI
    3.d Retorno de interrupción: Restaurar el estado del procesador

Examen 2019/20-2

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	20/06/2020	15:30

Examen 2019/20-2

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	20/06/2020	15:30

Examen 2019/20-2

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	20/06/2020	15:30

Examen 2019/20-2

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	20/06/2020	15:30

Examen 2019/20-2

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	20/06/2020	15:30

Examen 2019/20-2

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	20/06/2020	15:30