```
# Utilidades.s
# Archivo con distintas funciones utilizadas en Main.s
#
# Autores: Ka Fung 18-10492 & Christopher Gomez 18-10892
# Fecha: 25-nov-2021

# Funcion: Abre y lee un archivo dado.
# Entrada: $a0: Archivo.
#          $a1: Buffer.
#          $a2: Tamanio Buffer.
# Salida:  $v0: negativo si no logro leer archivo.
# Planificacion de registros:
# $t0: buffer
# $t1: archivo
leer_archivo:
    # Prologo
    sw   $fp,   ($sp)
    sw   $ra, -4($sp)
    move $fp,    $sp
    addi $sp,    $sp, -8

    move $t0, $a0   # Archivo
    move $t1, $a1   # Buffer
    move $t2, $a2   # Tamanio Buffer

    # Abrir archivo para leer
    li $v0, 13
    move $a0, $t0
    li $a1, 0
    syscall

    bltz $v0, leer_archivo_fin
    move $a0, $v0

    # Leer archivo
    li $v0, 14
    move $a1, $t1
    move $a2, $t2
    syscall

    bltz $v0, leer_archivo_fin

    add $t1, $t1, $v0
    sb $zero, ($t1) # Termina el buffer con un null

    # Cerrar el archivo
    li $v0, 16
    syscall

leer_archivo_fin:
    # Epilogo
    move $sp,    $fp
    lw   $fp,   ($sp)
    lw   $ra, -4($sp)

    jr $ra

```

```
 58
 59 # Funcion: Reserva memoria para guardar un dato dado.
 60 # Entrada: $a0: Buffer.
 61 #          $a1: Tamanio del dato.
 62 #          $a2: Booleano.
 63 #               0: itera hasta el tamanio del dato.
 64 #               1: itera hasta una comilla.
 65 # Salida:  $v0: Dir. del dato.
 66 #          $v1: Dir. del buffer actualizado.
 67 # Planificacion de registros:
 68 # $t0: buffer.
 69 # $t1: tamanio de memoria a reservar.
 70 # $t2: dir. del dato.
 71 # $t3: caracter actual.
 72 guardar_dato:
 73     # Prologo
 74     sw   $fp, ($sp)
 75     move $fp,  $sp
 76     addi $sp,  $sp, -4
 77
 78     move $t0, $a0
 79
 80     # Reserva memoria
 81     li  $v0, 9
 82     add $a0, $a1, 1
 83     syscall
 84
 85     bltz $v0, guardar_dato_fin
 86     move $t2, $v0
 87
 88     # Si se debe iterar hasta el tamanio del dato
 89     beqz $a2, guardar_hasta_tamanio
 90
 91     # Si se debe iterar hasta comilla
 92     guardar_hasta_comilla:
 93         lb $t3, ($t0)
 94
 95         # Si es una comilla o nulo, se termina
 96         beq  $t3, 34, guardar_dato_exitoso
 97         beqz $t3, guardar_dato_error
 98
 99         sb $t3, ($t2)
100
101         add $t0, $t0, 1
102         add $t2, $t2, 1
103
104         b guardar_hasta_comilla
105
106     guardar_hasta_tamanio:
107         lb $t3, ($t0)
108         sb $t3, ($t2)
109
110         add $t0, $t0,  1
111         add $t2, $t2,  1
112         add $a1, $a1, -1
113
114         bnez $a1, guardar_hasta_tamanio
```

```
115
116 guardar_dato_exitoso:
117     sb $zero, ($t2)
118
119     # Retorna dato y buffer
120     move $v1, $t0
121
122     b guardar_dato_fin
123
124 guardar_dato_error:
125     li $v0, -1
126
127 guardar_dato_fin:
128     # Epilogo
129     move $sp,  $fp
130     lw   $fp, ($sp)
131
132     jr $ra
133
134 # Funcion comparador.
135 # Compara dos strings a y b.
136 # Entrada: $a0: String a comparar.
137 #          $a1: String a comparar.
138 # Salida:  $v0: 0 si a < b,
139 #               1 de otra forma
140 # Planificacion de registros:
141 comparador:
142     # Prologo
143     sw   $fp, ($sp)
144     move $fp,  $sp
145     addi $sp,  $sp, -4
146
147     move $v0, $zero
148     comparador_loop:
149         lb $t0, ($a0)
150         lb $t1, ($a1)
151
152         # Si a < b, finaliza
153         blt $t0, $t1, comparador_fin
154
155         addi $a0, $a0, 1
156         addi $a1, $a1, 1
157
158         # Si a == b, revisar el siguiente
159         beq $t0, $t1, comparador_loop
160
161         # Si a > b, retorna -1
162         li $v0, 1
163
164 comparador_fin:
165     # Epilogo
166     move $sp,     $fp
167     lw   $fp,    ($sp)
168
169     jr $ra
170
171 # Funcion comparador carnets de una materia.
```

```
172 # Entrada: $a0: Par a comparar.
173 #          $a1: Par a comparar.
174 # Salida:  $v0: 0 si a<b,
175 #               1 de otra forma
176 comparador_carnet:
177     # Prologo
178     sw   $fp,   ($sp)
179     sw   $ra, -4($sp)
180     move $fp,    $sp
181     addi $sp,    $sp, -8
182 
183     lw $a0, ($a0)   # Estudiante a
184     lw $a0, ($a0)   # Carnet a
185 
186     lw $a1, ($a1)   # Estudiante b
187     lw $a1, ($a1)   # Carnet b
188 
189     jal comparador
190 
191 comparador_carnet_fin:
192     # Epilogo
193     move $sp,    $fp
194     lw   $fp,   ($sp)
195     lw   $ra, -4($sp)
196 
197     jr $ra
198 
199 
200 # Funcion comparador de solicitudes segun la modalidad:
201 # Prioridad a los estudiantes con menor numero de creditos aprobados.
202 # Entrada: $a0: Solicitud a comparar.
203 #          $a1: Solicitud a comparar.
204 # Salida:  $v0: 0 si a<b,
205 #               1 de otra forma
206 comparador_solicitud:
207     # Prologo
208     sw   $fp,   ($sp)
209     sw   $ra, -4($sp)
210     move $fp,    $sp
211     addi $sp,    $sp, -8
212 
213     lw $a0,   ($a0)   # Estudiante a
214     lw $a0, 12($a0)   # Creditos aprobados a
215 
216     lw $a1,   ($a1)   # Estudiante b
217     lw $a1, 12($a1)   # Creditos aprobados b
218 
219     move $v0, $zero
220 
221     # Si a < b, retorna 0
222     blt $a0, $a1, comparador_solicitud_fin
223 
224     # En cambio, retorna 1
225     li $v0, 1
226 
227 comparador_solicitud_fin:
228     # Epilogo
```

```
229     move $sp,    $fp
230     lw   $fp,   ($sp)
231     lw   $ra, -4($sp)
232 
233     jr $ra
234 
235 # Funcion limpiarBuffer.
236 # Limpia el buffer (redactar).
237 # Entrada: $a0: Dir. del buffer a limpiar.
238 #          $a1: Tamanio del buffer a comparar (multiplo de 4).
239 # Salida:  $v0: Dir. del buffer.
240 limpiarBuffer:
241     # Prologo
242     sw   $fp, ($sp)
243     move $fp,  $sp
244     addi $sp,  $sp, -4
245 
246     srl  $a1, $a1, 2 # Divide entre 4 el tamanio del buffer
247     move $v0, $a0
248     limpiarBuffer_ciclo:
249         beqz $a1, limpiarBuffer_fin
250 
251         sw $zero, ($a0)
252 
253         add $a0, $a0,  4
254         add $a1, $a1, -1
255         b limpiarBuffer_ciclo
256 
257 limpiarBuffer_fin:
258     # Epilogo
259     move $sp,     $fp
260     lw   $fp,    ($sp)
261 
262     jr $ra
263 
264 # Funcion atoi.
265 # Convierte un ASCII en entero.
266 # Entrada: $a0: ASCII.
267 #          $a1: Numero de caracteres a convertir.
268 # Salida:  $v0: Entero.
269 atoi:
270     # Prologo
271     sw   $fp, ($sp)
272     move $fp,  $sp
273     addi $sp,  $sp, -4
274 
275     move $v0, $zero
276     move $t0, $a0
277 
278     atoi_loop:
279         lb $t1, ($t0)
280 
281         add  $t0, $t0,  1
282         add  $a1, $a1, -1
283 
284         blt  $t1, 48, atoi_loop # Caracter es < '0'
285         bgt  $t1, 57, atoi_loop # Caracter es > '9'
```

```
286
287         # Se multiplica $v0*10
288         sll $t2, $v0, 3   # $t2 = 8  * $v0
289         sll $v0, $v0, 1   # $v0 = 2  * $v0
290         add $v0, $v0, $t2 # $v0 = 10 * $v0
291
292         # Se suma el digito
293         add $t1, $t1, -48
294         add $v0, $v0, $t1
295         bnez $a1, atoi_loop
296 atoi_fin:
297     # Epilogo
298     move $sp,     $fp
299     lw   $fp,    ($sp)
300
301     jr $ra
302
303 # Funcion itoa.
304 # Convierte un entero a ASCII.
305 # Entrada: $a0: Entero.
306 #          $a1: # Caracteres a convertir.
307 # Salida:  $v0: ASCII null-terminated.
308 #          $v1: Numero de caracteres escritos.
309 # Planificacion de registros:
310 # $t0: Caracter a agregar.
311 # $t1: -1 si el numero es negativo, 0 de otra forma.
312 # $t2: Signo '-' opcional si el numero es negativo.
313 # $t4: Copia del entero.
314 # $t5: Dir. del buffer (para iterar).
315 # $t6: 10.
316 itoa:
317     # Prologo
318     sw   $fp, ($sp)
319     move $fp,  $sp
320     addi $sp,  $sp, -4
321
322     move $t4, $a0
323     move $v1, $zero
324     move $t1, $zero
325
326     # Reserva memoria
327     li  $v0, 9
328     add $a0, $a1, 2
329     syscall
330
331     bltz $v0, itoa_fin
332     move $t5, $v0
333
334     bnez $t4,   itoa_distinto_de_cero  # Si el entero es distinto de cero
335     li   $t0,     '0'
336     sb   $t0,    ($t5)
337     sb   $zero, 1($t5)
338     add  $t5, $t5, -1
339     add  $v1, $v1,  1
340
341     b    itoa_fin
342
```

```
343 itoa_distinto_de_cero:
344     bgtz $t4, itoa_escribir_pos
345 
346     # Si es negativo, se agrega el signo y se convierte el entero a positivo
347     add $t5,  $t5, 1
348     abs $t4,  $t4
349     add $v1,  $v1, 1
350     li  $t1,  -1
351 
352 itoa_escribir_pos:
353     add $t5,    $t5, $a1
354     sb  $zero, ($t5) # Guarda nulo al final de la string
355     add $t5,    $t5, -1
356     li  $t6,    10
357 
358 itoa_loop:
359     # Se itera hasta que el entero = 0
360     div  $t4, $t6
361     mflo $t4       # $t4 = $a0 / 10
362     mfhi $t0       # $t0 = $a0 mod 10
363 
364     add $t0,  $t0,  48
365     sb  $t0, ($t5)    # Guarda el caracter en el buffer
366     add $t5,  $t5, -1
367     add $v1,  $v1, 1
368 
369     bnez $t4, itoa_loop
370 
371 itoa_fin:
372     add $v0,    $t5, 1
373 
374     li  $t2,  '-'
375     sb  $t2, ($t5)
376 
377     add $v0, $v0, $t1
378 
379     # Epilogo
380     move $sp,     $fp
381     lw   $fp,    ($sp)
382 
383     jr $ra
```