**Reporte de Proyecto**

**Equipo 4**

Natalia García A01191853

Paulina Escalante A01191962

David Benítez A01191731

Fabián Montemayor A01280156

**Datos Técnicos**

1. Hardware y Sistema Operativo.

Macbook Pro Retina 13 (Mid 2014)

Mac OS X Versión 10.11.1

2. Lenguaje de programación y versión.

Java SE 1.8.0\_40.

3. Tiempo de ciclo en computadora corrida (en MHz)

1600 MHz

4. Output de la corrida.

5. Código.

**Output del programa**

run:

Instrucción a procesar:

P 17 1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Se usaron los siguientes marcos de página:

0 1 2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Instrucción a procesar:

P 17 2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Se usaron los siguientes marcos de página:

3 4 5

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Instrucción a procesar:

P 17 3

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Se usaron los siguientes marcos de página:

6 7 8

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Instrucción a procesar:

A 10 1 0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Dirección real: 10

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Instrucción a procesar:

A 10 2 0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Dirección real: 34

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Instrucción a procesar:

A 10 2 0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Dirección real: 34

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Instrucción a procesar:

A 10 3 0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Dirección real: 58

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Instrucción a procesar:

A 11 3 0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Dirección real: 59

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Instrucción a procesar:

A 11 3 0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Dirección real: 59

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Instrucción a procesar:

A 11 3 0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Dirección real: 59

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Instrucción a procesar:

P 1983 4

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Se usaron los siguientes marcos de página:

0 9 10 11 12

13 14 15 16 17

18 19 20 21 22

23 24 25 26 27

28 29 30 31 32

33 34 35 36 37

38 39 40 41 42

43 44 45 46 47

48 49 50 51 52

53 54 55 56 57

58 59 60 61 62

63 64 65 66 67

68 69 70 71 72

73 74 75 76 77

78 79 80 81 82

83 84 85 86 87

88 89 90 91 92

93 94 95 96 97

98 99 100 101 102

103 104 105 106 107

108 109 110 111 112

113 114 115 116 117

118 119 120 121 122

123 124 125 126 127

128 129 130 131 132

133 134 135 136 137

138 139 140 141 142

143 144 145 146 147

148 149 150 151 152

153 154 155 156 157

158 159 160 161 162

163 164 165 166 167

168 169 170 171 172

173 174 175 176 177

178 179 180 181 182

183 184 185 186 187

188 189 190 191 192

193 194 195 196 197

198 199 200 201 202

203 204 205 206 207

208 209 210 211 212

213 214 215 216 217

218 219 220 221 222

223 224 225 226 227

228 229 230 231 232

233 234 235 236 237

238 239 240 241 242

243 244 245 246 247

248 249 250 251 252

253 254 255

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Instrucción a procesar:

A 1982 4 0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Dirección real: 2046

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Instrucción a procesar:

A 1982 4 0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Dirección real: 2046

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Instrucción a procesar:

P 16 5

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Se usaron los siguientes marcos de página:

1 2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Instrucción a procesar:

F

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Proceso 1 cargado en memoria.

Proceso 2 cargado en memoria.

Proceso 3 cargado en memoria.

Proceso 4 cargado en memoria.

Proceso 5 cargado en memoria.

Procesos terminados: 0

Page faults: 0

Swap ins: 0

Swap outs: 3

Turnaround promedio: 0.0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Instrucción a procesar:

P 2032 1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Se usaron los siguientes marcos de página:

0 1 2 3 4

5 6 7 8 9

10 11 12 13 14

15 16 17 18 19

20 21 22 23 24

25 26 27 28 29

30 31 32 33 34

35 36 37 38 39

40 41 42 43 44

45 46 47 48 49

50 51 52 53 54

55 56 57 58 59

60 61 62 63 64

65 66 67 68 69

70 71 72 73 74

75 76 77 78 79

80 81 82 83 84

85 86 87 88 89

90 91 92 93 94

95 96 97 98 99

100 101 102 103 104

105 106 107 108 109

110 111 112 113 114

115 116 117 118 119

120 121 122 123 124

125 126 127 128 129

130 131 132 133 134

135 136 137 138 139

140 141 142 143 144

145 146 147 148 149

150 151 152 153 154

155 156 157 158 159

160 161 162 163 164

165 166 167 168 169

170 171 172 173 174

175 176 177 178 179

180 181 182 183 184

185 186 187 188 189

190 191 192 193 194

195 196 197 198 199

200 201 202 203 204

205 206 207 208 209

210 211 212 213 214

215 216 217 218 219

220 221 222 223 224

225 226 227 228 229

230 231 232 233 234

235 236 237 238 239

240 241 242 243 244

245 246 247 248 249

250 251 252 253

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Instrucción a procesar:

P 17 2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Se usaron los siguientes marcos de página:

0 254 255

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Instrucción a procesar:

A 10 2 0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Dirección real: 2034

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Instrucción a procesar:

P 16 3

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Se usaron los siguientes marcos de página:

1 2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Instrucción a procesar:

A 7 1 0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Dirección virtual: 7

Página 3 del proceso 1 transferida a memoria secundaria

Dirección virtual asignada: 0

Página 0 del proceso 1 transferida a memoria principal

Dirección real: 27

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Instrucción a procesar:

A 15 1 0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Dirección virtual: 23

Página 0 del proceso 1 transferida a memoria secundaria

Dirección virtual asignada: 18

Página 1 del proceso 1 transferida a memoria principal

Dirección real: 27

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Instrucción a procesar:

A 8 1 0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Dirección real: 24

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Instrucción a procesar:

L 2

Liberar Proceso 2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Marcos liberados de memoria principal:

0 254

255

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Marcos liberados de memoria secundaria:

SIN MODIFICACIÓN

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Instrucción a procesar:

P 16 4

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Se usaron los siguientes marcos de página:

0 254

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Instrucción a procesar:

A 10 4 0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Dirección real: 2034

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Instrucción a procesar:

L 1

Liberar Proceso 1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Marcos liberados de memoria principal:

3 4

5 6 7 8 9

10 11 12 13 14

15 16 17 18 19

20 21 22 23 24

25 26 27 28 29

30 31 32 33 34

35 36 37 38 39

40 41 42 43 44

45 46 47 48 49

50 51 52 53 54

55 56 57 58 59

60 61 62 63 64

65 66 67 68 69

70 71 72 73 74

75 76 77 78 79

80 81 82 83 84

85 86 87 88 89

90 91 92 93 94

95 96 97 98 99

100 101 102 103 104

105 106 107 108 109

110 111 112 113 114

115 116 117 118 119

120 121 122 123 124

125 126 127 128 129

130 131 132 133 134

135 136 137 138 139

140 141 142 143 144

145 146 147 148 149

150 151 152 153 154

155 156 157 158 159

160 161 162 163 164

165 166 167 168 169

170 171 172 173 174

175 176 177 178 179

180 181 182 183 184

185 186 187 188 189

190 191 192 193 194

195 196 197 198 199

200 201 202 203 204

205 206 207 208 209

210 211 212 213 214

215 216 217 218 219

220 221 222 223 224

225 226 227 228 229

230 231 232 233 234

235 236 237 238 239

240 241 242 243 244

245 246 247 248 249

250 251 252 253

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Marcos liberados de memoria secundaria:

0 1 2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Instrucción a procesar:

L 2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

No existe el proceso 2 en memoria

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Instrucción a procesar:

L 3

Liberar Proceso 3

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Marcos liberados de memoria principal:

1 2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Marcos liberados de memoria secundaria:

SIN MODIFICACIÓN

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Instrucción a procesar:

F

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Turnaround de proceso 1: 7ms

Turnaround de proceso 2: 2ms

Turnaround de proceso 3: 6ms

Proceso 4 cargado en memoria.

Procesos terminados: 3

Page faults: 0

Swap ins: 2

Swap outs: 5

Turnaround promedio: 5.0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Tamaño excede espacio en memoria (2048):

4000

Error de linea:

P 4000 1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Instrucción a procesar:

F

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Procesos terminados: 0

Page faults: 0

Swap ins: 0

Swap outs: 0

Turnaround promedio: 0.0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Instrucción a procesar:

P 10 1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Se usaron los siguientes marcos de página:

0 1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Instrucción a procesar:

A 15 1 0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Dirección real: 15

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Instrucción a procesar:

L 1

Liberar Proceso 1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Marcos liberados de memoria principal:

0 1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Marcos liberados de memoria secundaria:

SIN MODIFICACIÓN

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Instrucción a procesar:

F

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Turnaround de proceso 1: 1ms

Procesos terminados: 1

Page faults: 0

Swap ins: 0

Swap outs: 0

Turnaround promedio: 1.0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Linea de pruebas invalida:

Z

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Instrucción a procesar:

F

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Procesos terminados: 0

Page faults: 0

Swap ins: 0

Swap outs: 0

Turnaround promedio: 0.0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Instrucción a procesar:

A 15 1 0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

La página no está cargada en memoria

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Instrucción a procesar:

F

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Procesos terminados: 0

Page faults: 0

Swap ins: 0

Swap outs: 0

Turnaround promedio: 0.0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Instrucción a procesar:

P 10 1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Se usaron los siguientes marcos de página:

0 1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Error de sintaxis de linea:

A ocho 1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Instrucción a procesar:

L 1

Liberar Proceso 1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Marcos liberados de memoria principal:

0 1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Marcos liberados de memoria secundaria:

SIN MODIFICACIÓN

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Instrucción a procesar:

F

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Turnaround de proceso 1: 1ms

Procesos terminados: 1

Page faults: 0

Swap ins: 0

Swap outs: 0

Turnaround promedio: 1.0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Instrucción a procesar:

P 10 5328000

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Se usaron los siguientes marcos de página:

0 1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Instrucción a procesar:

A 2 5328000 0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Dirección real: 2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Instrucción a procesar:

L 5328000

Liberar Proceso 5328000

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Marcos liberados de memoria principal:

0 1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Marcos liberados de memoria secundaria:

SIN MODIFICACIÓN

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Instrucción a procesar:

F

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Turnaround de proceso 5328000: 1ms

Procesos terminados: 1

Page faults: 0

Swap ins: 0

Swap outs: 0

Turnaround promedio: 1.0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Instrucción a procesar:

E

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Terminar pruebas

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

BUILD SUCCESSFUL (total time: 0 seconds)

**Programa en Java (Main)**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package Main;

import Classes.\*;

/\*\*

\*

\* @author Equipo4

\*/

public class ProyectoFinalOperativos\_Equipo4 {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

// TODO code application logic here

//Lee archivo de prueba

Controller control = new Controller("Pruebas1.txt");

control.leerArchivo();

}

}

**Programa en Java (Controller)**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package Classes;

import java.io.File;

import java.io.IOException;

import java.util.Arrays;

import java.util.LinkedList;

import java.util.Scanner;

/\*\*

\*

\* @author Equipo4

\*/

public class Controller {

// Contiene memorias y hace handling de comandos ACCESAR, AGREGAR, etc.

EventHandler handler;

// Contiene procesos como queue

LinkedList<Process> processList;

// Nombre de archivo

String fileName;

// Scanner de archivo

Scanner scan;

// ID de proceso entrante

Integer pID;

// Tama√±o del proceso

Integer pSize;

// Comando en archivo

String[] commandLine;

// Caracter de comando

char charCommand;

// Direccion asignada al proceso

int address;

// Datos se guardan para resumen del final

Summary summary;

// Modified bit

boolean bitMod;

// Controller constructor

public Controller(String fName) {

// Asigna valores default

this.handler = new EventHandler();

this.processList = new LinkedList();

this.fileName = fName;

this.pID = -1;

this.pSize = 0;

this.commandLine = null;

this.charCommand = '\0';

this.address = 0;

this.summary = new Summary();

}

//lee archivo de pruebas, valida inputs

public void leerArchivo() {

// Variable boleana en pasado

boolean fileRead = false;

String line;

// Leer archivo

try {

scan = new Scanner(new File(fileName));

fileRead = true;

}

catch (IOException e) {

System.out.println("Archivo no existe");

}

//Si fue leido exitosamente

if (fileRead) {

// Leer linea

line = scan.nextLine();

line = line.trim();

if (!line.equals("")) {

}

// Leer archivo hasta que sea E

while (!line.equals("E")) {

if (!line.equals("")) {

// Leer primera palabra

commandLine = line.split(" ");

charCommand = commandLine[0].charAt(0);

switch(charCommand) {

// Agregar proceso

case 'P':

if(validateP(line)) {

System.out.println("Instrucci√≥n a procesar:");

System.out.println(line);

System.out.println();

createP();

}

break;

// Accesar proceso

case 'A':

if(validateA(line)) {

System.out.println("Instrucci√≥n a procesar:");

System.out.println(line);

createA();

System.out.println();

}

break;

// Liberar proceso

case 'L':

if(validateL(line)) {

System.out.println("Instrucci√≥n a procesar:");

System.out.println(line);

System.out.println();

createL();

}

break;

// Fin de acciones

case 'F':

System.out.println("Instrucci√≥n a procesar:");

System.out.println(line);

System.out.println();

createF();

break;

default:

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("Linea de pruebas invalida:");

System.out.println(line);

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println();

break;

}

}

// Read new line

line = scan.nextLine();

line = line.trim();

}

System.out.println("Instrucci√≥n a procesar:");

System.out.println(line);

System.out.println();

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("Terminar pruebas");

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.exit(0);

// Finish reading

scan.close();

}

}

// Crea nuevo evento de agregar proceso a memoria

private void createP() {

// Se crea proceso

Process newProcess = new Process(pID, pSize);

// Se carga al handler

if(handler.load(newProcess, summary)) {

// Se carga a la lista

processList.add(newProcess);

}

}

// Validar file input de P

private boolean validateP(String line) {

// Palabras sin espacio, checar sintaxis

if (line.trim().split("\\s+").length != 3) {

// Error de sintaxis

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("Error de sintaxis de linea:");

System.out.println(line);

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println();

return false;

}

// Checar si palabras son numeros o letras

try {

pSize = Integer.parseInt(commandLine[1]);

// Checar tama√±o de letra

if (pSize > 2048) {

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("Tama√±o excede espacio en memoria (2048):");

System.out.println(commandLine[1]);

System.out.println("Error de linea:");

System.out.println(line);

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println();

return false;

}

}

catch(NumberFormatException e) {

//se imprime el tipo de error

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("Error de sintaxis de linea:");

System.out.println(line);

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println();

return false;

}

// Checar si palabras son numeros o letras

try {

pID = Integer.parseInt(commandLine[2]);

}

catch(NumberFormatException e) {

//se imprime el tipo de error

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("Error de sintaxis de linea:");

System.out.println(line);

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println();

return false;

}

// Bulletproof input

return true;

}

// Crear nuevo evento de accesar datos

private void createA() {

// Accesar proceso

handler.access(address, pID, bitMod, processList, summary);

}

// Validar file input de A

private boolean validateA(String line) {

// Palabras sin espacio, checar sintaxis

if (line.trim().split("\\s+").length != 4) {

// Error de sintaxis

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("Error de sintaxis de linea:");

System.out.println(line);

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println();

return false;

}

// Checar si palabras son numeros o letras

try {

address = Integer.parseInt(commandLine[1]);

}

catch(NumberFormatException e) {

//se imprime el tipo de error

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("Error de sintaxis de linea:");

System.out.println(line);

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println();

return false;

}

// Checar si palabras son numeros o letras

try {

pID = Integer.parseInt(commandLine[2]);

}

catch(NumberFormatException e) {

//se imprime el tipo de error

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("Error de sintaxis de linea:");

System.out.println(line);

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println();

return false;

}

// Validate bit

switch (commandLine[3]) {

case "0":

bitMod = false;

break;

case "1":

bitMod = true;

break;

default:

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("Error de sintaxis de linea:");

System.out.println(line);

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println();

return false;

}

// Bulletproof input

return true;

}

// Crear nuevo evento de liberar proceso

private void createL() {

// Liberar proceso

handler.removeProcess(pID, summary, processList);

}

// Validar file input de L

private boolean validateL(String line) {

// Palabras sin espacio, checar sintaxis

if (line.trim().split("\\s+").length != 2) {

// Error de sintaxis

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("Error de sintaxis de linea:");

System.out.println(line);

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println();

return false;

}

// Checar si palabras son numeros o letras

try {

pID = Integer.parseInt(commandLine[1]);

}

catch(NumberFormatException e) {

//se imprime el tipo de error

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("Error de sintaxis de linea:");

System.out.println(line);

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println();

return false;

}

return true;

}

// Nuevo evento de Fin de lectura de datos agrupados

private void createF() {

// Finish

handler.end(processList, summary);

// Reinicia variables

summary = new Summary();

handler = new EventHandler();

processList = new LinkedList();

}

}

**Programa en Java (EventHandler)**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package Classes;

import java.util.Calendar;

import java.util.ArrayList;

import java.util.LinkedList;

import java.util.List;

import java.util.Queue;

/\*\*

\*

\* @author Equipo4

\*/

public class EventHandler {

// Arreglo de marcos de p√°gina en memoria principal.

private Frame[] mainMemory;

// Arreglo de marcos de p√°gina en memoria secundaria.

private Frame[] secondaryMemory;

// Queue para manejar FIFO.

private LinkedList<Integer> mainMemoryQueue;

private LinkedList<Integer> secondaryMemoryQueue;

// Arreglo en posicion 0 espacios libres en memoria principal, indice 1

// libres en memoria secundaria

private Integer[] frameAvailability;

private Summary report;

private LinkedList<Integer> mainMemoryFrameAvailability;

private LinkedList<Integer> secondaryMemoryFrameAvailability;

public EventHandler() {

secondaryMemoryFrameAvailability = new LinkedList<Integer>();

mainMemoryFrameAvailability = new LinkedList<Integer>();

mainMemory = new Frame[256];

secondaryMemory = new Frame[512];

mainMemoryQueue = new LinkedList<Integer>();

secondaryMemoryQueue = new LinkedList<Integer>();

frameAvailability = new Integer[2];

frameAvailability[0] = 0;

frameAvailability[1] = 0;

for (int i = 0; i < 256; i++) {

mainMemory[i] = new Frame();

secondaryMemory[i] = new Frame();

}

for (int i = 256; i < 512; i++) {

secondaryMemory[i] = new Frame();

}

}

public void moveToSecondaryMemory (Process p, int spaceToMove) {

int processID, pageNumber, frameNumber;

if (!hasEnoughSpace(spaceToMove, 1, secondaryMemory)) {

//si no hay espacio, hace espacio con freeSpace

freeSpace (spaceToMove,

secondaryMemoryFrameAvailability, 1,

secondaryMemoryQueue);

}

for (int i = 0; i < spaceToMove &&

mainMemory[mainMemoryFrameAvailability.peekLast()].getProcessID() != -1; i++) {

report.swapsOut++;

frameNumber = mainMemoryFrameAvailability.pollLast();

mainMemoryFrameAvailability.addFirst(frameNumber);

processID = mainMemory[frameNumber].getProcessID();

pageNumber = mainMemory[frameNumber].getPageNumber();

mainMemory[frameNumber].setProcessID(-1);

frameNumber = secondaryMemoryFrameAvailability.pollFirst();

secondaryMemory[frameNumber].setProcessID(processID);

secondaryMemory[frameNumber].setPageNumber(pageNumber);

secondaryMemoryQueue.add(frameNumber);

}

}

/\*\*

\* Mueve una pagina de memoria secundaria a principal

\*

\* @param SM\_pageNumber Es el numero de pagina del proceso

\* @param SM\_processID Es el id del proceso

\* @param SM\_frame Es el frame de memoria secundaria donde est√° la pagina

\*/

public void movePageToPrimaryMemory(int SM\_pageNumber, int SM\_processID, int SM\_frame){

//checa que haya un marco libre en memoria principal,

//se actualiza la lista de libres en principal

boolean fitsInMM = hasEnoughSpace(1, 0, mainMemory);

int MM\_pageNumber, MM\_processID, MM\_frame;

//si no hay espacio, hacer swaps

if(!fitsInMM){

//guardo los valores de la pagina actualmente en memoria principal

MM\_frame = mainMemoryQueue.poll();

MM\_pageNumber = mainMemory[MM\_frame].getPageNumber();

MM\_processID = mainMemory[MM\_frame].getProcessID();

//actualizo los valores por los de la memoria secundaria

mainMemory[MM\_frame].setPageNumber(SM\_pageNumber);

mainMemory[MM\_frame].setProcessID(SM\_processID);

//actualizo valores por los de la memoria principal

secondaryMemory[SM\_frame].setPageNumber(MM\_pageNumber);

secondaryMemory[SM\_frame].setProcessID(MM\_processID);

//Se movio a que frame de memoria principal?

System.out.println("P√°gina " + MM\_pageNumber + " del proceso "

+ MM\_processID + " transferida a memoria secundaria");

System.out.println("Direcci√≥n virtual asignada: " + (SM\_frame%8 + SM\_frame\*8));

//Se movio a que frame de memoria principal?

System.out.println("P√°gina " + SM\_pageNumber + " del proceso "

+ SM\_processID + " transferida a memoria principal");

System.out.println("Direcci√≥n real: " + (MM\_frame%8 + MM\_frame\*8));

mainMemoryQueue.push(MM\_frame);

//secondaryMemoryQueue.push(SM\_frame); //buscar y sacar este frame??

report.swapsOut++;

}

//si hay espacio, solo mover a principal

else{

MM\_frame = mainMemoryFrameAvailability.peek();

//actualizo los valores por los de la memoria secundaria

mainMemory[MM\_frame].setPageNumber(SM\_pageNumber);

mainMemory[MM\_frame].setProcessID(SM\_processID);

//Se movio a que frame de memoria principal?

System.out.println("P√°gina " + SM\_pageNumber + " del proceso "

+ SM\_processID + " transferida a memoria principal");

System.out.println("Direcci√≥n real: " + (MM\_frame%8 + MM\_frame\*8));

//se borran datos de pagina del marco en la memoria secundaria

secondaryMemory[SM\_frame].setPageNumber(0);

secondaryMemory[SM\_frame].setProcessID(-1);

mainMemoryQueue.push(MM\_frame);

//quitar secondary de queue?

}

report.swapsIn++;

}

//sobreescribe main memories availability

//type = 0 para principal

//type = 1 para secundaria

public boolean hasEnoughSpace (int spaceNeeded, int type,

Frame[] frameArray) {

LinkedList<Integer> memoryFrameAvailability = new LinkedList<Integer>();

int max;

//tipo = 0 para principal

if (type == 0) {

max = 256;

}

//tipo = 1 para secundaria

else {

max = 512;

}

//itera hasta pasarse de marcos o llenar la demanda de libres

for (int i = 0; i < max && frameAvailability[type] != spaceNeeded; i++) {

if (frameArray[i].getProcessID() == -1) {

frameAvailability[type]++;

memoryFrameAvailability.add(i);

}

}

//copia los libres a la lista correspondiente

if (type == 0) {

mainMemoryFrameAvailability = memoryFrameAvailability;

}

else {

secondaryMemoryFrameAvailability = memoryFrameAvailability;

}

return (frameAvailability[type] >= spaceNeeded);

}

// M√©todo que muestra los datos del summary despu√©s de un conjunto de

// instrucciones.

// Recibe la lista de procesos y el resumen.

public void end(LinkedList<Process> processList, Summary summary) {

// Revisa los procesos que no terminaron y muestra el error.

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

for (Process process : processList) {

if (process.getEndTime() == null) {

System.out.println("Proceso " + process.getId() +

" cargado en memoria.");

}

else {

System.out.println("Turnaround de proceso " + process.getId() +

": " + process.getTurnaround() + "ms");

}

}

System.out.println("Procesos terminados: " +

summary.getTerminatedProcesses());

System.out.println("Page faults: "+ summary.getPageFaults());

System.out.println("Swap ins: " + summary.getSwapsIn());

System.out.println("Swap outs: "+ summary.getSwapsOut());

System.out.println("Turnaround promedio: " +

summary.getAverageTurnaround());

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println();

}

public boolean access(int address, int pID, boolean bitMod, LinkedList<Process>

processList, Summary summary) {

int pageNumber = address/8;

frameAvailability[0] = 0;

frameAvailability[1] = 0;

//innecesario, o cambiar a que regrese index y su tipo de memoria

if (!isLoaded(pID)) {

//erorr no esta cargado

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("La p√°gina no est√° cargada en memoria");

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println();

return false;

}

boolean pageFound = false;

boolean pageFoundInSecondaryM = false;

System.out.println();

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

//checar si esta cargada en memoria principal

for(int i = 0; i < 256; i++){

if(mainMemory[i].getProcessID() == pID &&

mainMemory[i].getPageNumber() == pageNumber){

System.out.println("Direcci√≥n real: " + (address%8 + i\*8));

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

//pageFound = true;

return true;

}

}

//si no lo encuentra, buscar en memoria secundaria

if(!pageFound){

for(int i = 0; i < 512; i++){

//lo encuentra

if(secondaryMemory[i].getProcessID() == pID &&

secondaryMemory[i].getPageNumber() == pageNumber){

System.out.println("Direcci√≥n virtual: " + (address%8 + i\*8));

//pasa de secundaria a principal

movePageToPrimaryMemory(pageNumber, pID, i);

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

//pageFoundInSecondaryM = true;

return true;

}

}

}

if(!pageFoundInSecondaryM){

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("La p√°gina no est√° cargada en memoria");

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println();

}

return false;

}

// M√©todo removeProcess que libera un proceso de memoria.

// Recibe de par√°metros el processId, el summary y la lista de procesos.

// El m√©todo calcula el turnaround y muestra los marcos liberados.

public void removeProcess(int pID, Summary summary,

LinkedList<Process> processList) {

boolean exists = false; // Variable que guarda si el proceso existe.

// Checa que el proceso exista

for (Process process : processList) {

if (process.getId() == pID && !process.getEnd()) {

exists = true;

}

}

// Si el proceso existe, se libera.

if (exists) {

System.out.println("Liberar Proceso " + pID);

// Tiempo de llegada

Calendar arrivalTime = null;

// Proceso temporal

Process p = null;

// Index in list

int processIndex = 0;

// Saca el tiempo de llegada y el index.

for (Process process : processList) {

if (process.getId() == pID) {

arrivalTime = process.getArrivalTime();

// Guardar indice

processIndex = processList.indexOf(process);

}

}

// Get actual time

Calendar terminationTime = Calendar.getInstance();

terminationTime.getTime();

// Turnaround calculations

long turnaround = (terminationTime.getTimeInMillis() -

arrivalTime.getTimeInMillis());

processList.get(processIndex).setEndTime(terminationTime);

summary.updateTerminatedProcesses();

summary.updateTotalTurnaround(turnaround);

processList.get(processIndex).getTurnaround();

processList.get(processIndex).setEnd(true);

int countP = 0;

int countS = 0;

// Revisa memorias y borra proceso con pID

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("Marcos liberados de memoria principal: ");

for (int i = 0; i < 256; i++) {

if (mainMemory[i].getProcessID() == pID) {

countP++;

mainMemory[i].setProcessID(-1);

// Encontrar y borrar en queue

mainMemoryQueue.remove((Object)i);

System.out.print(i + "\t \t");

if ((i+1) % 5 == 0) {

System.out.println();

}

}

}

if (countP == 0) {

System.out.println("SIN MODIFICACI√ìN");

}

else {

System.out.println();

}

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("Marcos liberados de memoria secundaria: ");

for (int i = 0; i < 512; i++) {

if (secondaryMemory[i].getProcessID() == pID) {

countS++;

secondaryMemory[i].setProcessID(-1);

// Encontrar y borrar en queue

secondaryMemoryQueue.remove((Object)i);

System.out.print(i + "\t \t");

if ((i+1) % 5 == 0) {

System.out.println();

}

}

}

if (countS == 0) {

System.out.println("SIN MODIFICACI√ìN");

}

else {

System.out.println();

}

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println();

}

else {

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("No existe el proceso " + pID + " en memoria");

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println();

}

report = summary;

}

public boolean isLoaded (int pID) {

for (int i = 0; i < 256; i++) {

if (mainMemory[i].getProcessID() == pID ||

secondaryMemory[i].getProcessID() == pID) {

return true;

}

}

for (int i = 256; i < 512; i++) {

if (secondaryMemory[i].getProcessID() == pID) {

return true;

}

}

return false;

}

public void freeSpace (int spaceNeeded, LinkedList<Integer> memoryFrameAvailability, int type,

Queue<Integer> tempQueue) {

int frameNumber = 0;

for (int i = 0; i < spaceNeeded; i++) {

frameNumber = tempQueue.poll();

memoryFrameAvailability.add(frameNumber);

frameAvailability[type]++;

}

if (type == 0) {

mainMemoryFrameAvailability = memoryFrameAvailability;

}

else {

secondaryMemoryFrameAvailability = memoryFrameAvailability;

}

}

public boolean load (Process p, Summary r) {

report = r;

boolean fitsInMainMemory = false;

frameAvailability[0] = 0;

frameAvailability[1] = 0;

int pageNum = p.getSize()/8;

if (p.getSize() % 8 != 0) {

pageNum++;

}

p.setPageNumber(pageNum);

if (isLoaded(p.getId())) {

// Checa si el proceso ya existe

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("Este proceso ya est√° cargado en memoria");

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println();

return false;

}

else {

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("Se usaron los siguientes marcos de p√°gina: ");

if (!hasEnoughSpace (p.getPageNumber(), 0, mainMemory)) {

int tmp = frameAvailability[0];

//liberar espacio

freeSpace(p.getPageNumber()-frameAvailability[0],

mainMemoryFrameAvailability, 0, mainMemoryQueue);

moveToSecondaryMemory(p, p.getPageNumber()-tmp);

}

//solo se carga en memoria

int frameNumber = 0;

int pageNumber = 0;

for (int i = 0; i < frameAvailability[0]; i++) {

frameNumber = mainMemoryFrameAvailability.get(i);

mainMemory[frameNumber].setProcessID(p.getId());

mainMemory[frameNumber].setPageNumber(pageNumber);

System.out.print(frameNumber + "\t \t");

pageNumber++;

mainMemoryQueue.add(frameNumber);

if ((i+1) % 5 == 0) {

System.out.println();

}

}

System.out.println();

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

mainMemoryFrameAvailability.clear();

System.out.println();

}

return true;

}

}

**Programa en Java (Frame)**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package Classes;

/\*\*

\* Clase Frame, guarda la informaci√≥n de un frame y su process

\* @author Equipo4

\*/

public class Frame{

private int size; // size del frame

private int processID; // id del process dentro del frame

private int pageNumber; // n√∫mero de p√°gina

//Constructor de la clase Frame

public Frame(){

this.size = 8;

this.processID = -1;

this.pageNumber = 0;

}

//Getters

public int getSize(){

return this.size;

}

public int getProcessID(){

return this.processID;

}

public int getPageNumber(){

return this.pageNumber;

}

//Setters

public void setSize(int size){

this.size = size;

}

public void setProcessID(int processID){

this.processID = processID;

}

public void setPageNumber(int pageNumber){

this.pageNumber = pageNumber;

}

};

**Programa en Java (Page)**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package Classes;

/\*\*

\* Page

\* Class <code>Page</code>, stores page information

\* @author Team4

\*/

public class Page {

private int frame; // El n√∫mero de frame en donde se encuentra la p√°gina.

private boolean residenceBit; // El bit de residencia.

public Page() {

residenceBit = false;

}

// M√©todos para accesar la informaci√≥n de la p√°gina.

public int getFrame() {

return frame;

}

public boolean getResidenceBit() {

return residenceBit;

}

// M√©todos para modificar la informaci√≥n de la p√°gina.

public void setFrame(int frame) {

this.frame = frame;

}

public void setResidenceBit(boolean residenceBit) {

this.residenceBit = residenceBit;

}

}

**Programa en Java (Process)**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package Classes;

import java.util.Calendar;

/\*\*

\* Process

La clase <code>Process</code> guarda toda la informaci√≥n de un proceso, es un

\* simulador de una PCB.

\* @author Equipo4

\*/

public class Process {

private int id; // Es el id del proceso.

private int pageNumber; // N√∫mero de p√°ginas que ocupa el proceso.

private int size; // Tama√±o del proceso.

// Arreglo de las p√°ginas ocupadas por el proceso.

private Page[] arrPages;

private Calendar arrivalTime; // Tiempo de llegada del proceso.

private long turnaround; // Tiempo de turnaround del proceso.

private Calendar endTime; // Tiempo de terminaci√≥n del proceso.

private boolean end;

public Process(int id, int size) {

this.id = id;

this.size = size;

if(this.size%8 != 0) {

this.pageNumber = size/8 + 1;

}

else {

this.pageNumber = size/8;

}

this.arrPages = new Page[pageNumber];

this.arrivalTime = Calendar.getInstance();

this.arrivalTime.getTime();

this.turnaround = 0;

this.endTime = null;

this.end = false;

}

// Metodos para obtener los datos del proceso

public long getTurnaround() {

if(endTime != null)

{

turnaround = (endTime.getTimeInMillis() - arrivalTime.getTimeInMillis());

return (endTime.getTimeInMillis() - arrivalTime.getTimeInMillis());

}

return 0;

}

public Calendar getEndTime() {

return endTime;

}

public int getId() {

return id;

}

public int getPageNumber() {

return pageNumber;

}

public boolean getEnd() {

return end;

}

public void setEnd(boolean value) {

this.end = value;

}

public int getSize() {

return size;

}

public Page[] getArrPages() {

return arrPages;

}

public Calendar getArrivalTime() {

return arrivalTime;

}

// Metodos para modificar los datos del proceso

public void setPageNumber(int pageNumber) {

this.pageNumber = pageNumber;

}

public void setEndTime(Calendar endTime) {

this.endTime = endTime;

this.end = true;

}

}

**Programa en Java (Summary)**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package Classes;

/\*\*

\* Summary

\* La clase <code>Summary</code> guarda toda la informaci√≥n que se imprime en el

\* reporte final con la instrucci√≥n F.

\* @author Equipo 4

\*/

class Summary {

// Procesos terminados en el conjunto de instrucciones.

int terminatedProcesses;

// Cantidad de fallos de p√°gina en el conjunto de instrucciones.

int pageFaults;

// Cantidad de swaps in de p√°gina en el conjunto de instrucciones.

int swapsIn;

// Cantidad de swaps out de p√°gina en el conjunto de instrucciones.

int swapsOut;

// Tiempo de turnaround total de todos los procesos.

long totalTurnaround;

// Tiempo de turnaround promedio de los procesos.

double averageTurnaround;

public Summary() {

// Se inicializan todas las variables en 0.

this.terminatedProcesses = 0;

this.pageFaults = 0;

this.swapsIn = 0;

this.swapsOut = 0;

this.totalTurnaround = 0;

this.averageTurnaround = 0;

}

// M√©todos para accesar las variables.

public int getTerminatedProcesses() {

return terminatedProcesses;

}

public int getPageFaults() {

return pageFaults;

}

public int getSwapsIn() {

return swapsIn;

}

public int getSwapsOut() {

return swapsOut;

}

public long getTotalTurnaround() {

return totalTurnaround;

}

public double getAverageTurnaround() {

if (terminatedProcesses > 0) {

return totalTurnaround/terminatedProcesses;

}

return 0;

}

// M√©todos para modificar las variables.

// Agrega 1 a los procesos terminados.

public void updateTerminatedProcesses() {

terminatedProcesses += 1;

}

public void updatePageFaults() {

pageFaults += 1;

}

public void updateSwapsIn() {

swapsIn += 1;

}

public void updateSwapsOut() {

swapsOut += 1;

}

public void updateTotalTurnaround(long turnaround) {

totalTurnaround += turnaround;

}

}