ASSEMBLY FILE 4:

; Author: Lachlan Forgan

; Assignment: HW1 Assembly Lang Program

; Due Date: 4/2025

; Description: Allocate 100 words, fill with a mathematical sequence, free memory

function

Counter long 100 ; Counter for counting 100 iterations

Addr long 0 ; Holds address of dynamically allocated memory

R long 13 ; Arbitrary number R for filling first word

-M long -3 ; Arbitrary number M for performing operations

Temp long 0 ; Temp for holding word value after moving next word

Start: mov R2, 100 ; Move 100 into R2

SystemCall 4 ; System call memory alloc: size of 100 words in R2

mov Addr, R1 ; R1 holds address of allocated memory, put in Addr

mov R4, Addr ; Put addr in R4 for deferred referencing

mov [R4], R ; Copy 4 into first of dynamically allocated memory, address is in R4

mov Temp, R4 ; Move R4 into temp for use in next word

add R4, 1 ; Increment R4 by 1

sub Counter, 1 ; Decrease counter by 1

Loop: mov [R4], Temp ; Copy previous word into new word

mul [R4], M ; Multiply word by M

sub Counter, 1 ; Decrease counter by 1

mov Temp, [R4] ; copy value at address in R4 into temp

add R4, 1 ; increment R4 (go to next memory location)

BrOnPlus Counter, Loop ; If counter > 0, go to Loop

mov R1, addr ; Copy Addr into R1

mov R2, 100 ; Set R2 to 100 (for freeing memory)

SystemCall 5 ; MemFreeSystemCall

stop Halt ; halt

END Start ; end of program, has value for start

MACHINE LANG FILE 4:

Free 100 words in memory, fill them with different numbers using a sequence and operations, then free the memory and halt.

140 100 // Counter

141 0 // Addr

142 13 // R - random, arbitrary number

143 -3 // -M where M = 3

144 0 // Temp

145 51260 // Start - move 100 into R2

146 100

147 120000 // System call: free 100 words

148 4 // SystemCallID

149 55011 // Move word address start from R1 -> Addr

150 141

151 51450 // Copy Addr into R4

152 141 // Addr

153 52450 // Move R into address in R4

154 142 // R

155 55024 // Copy value in R4 into temp

156 144 // Temp

157 11460 // Increase R4 by 1

158 1

159 25060 // Decrease Counter by 1

160 140 // Counter

161 1

162 52450 // LOOP: Copy previous value (temp) into new R4 address

163 144

164 32450 // multiply value at this address by M

165 143

166 25060 // decrease counter by 1

167 140 // Counter

168 1

169 55024 // Copy value at address in R4 into temp

170 144

171 11460 // Increase address in R4 by 1

172 1

173 85050 // BRANCH if Counter > 0

174 140 // Counter

175 162 // loop

176 51150 // Mov Addr into R1

177 141 // Addr

178 51260 // Mov size (100) into R2

179 100

180 120000 // System call

181 5 // MemFreeSystemCall

182 000000 // Stop - halt

-1 145 // Start

Symbol Table:

|  |  |
| --- | --- |
| LABEL | ADDRESS |
| Counter | 140 |
| Addr | 141 |
| R | 142 |
| -M | 143 |
| Temp | 144 |
| Start | 145 |
| LOOP | 162 |
| Stop | 182 |

ASSEMBLY FILE 5:

; Author: Lachlan Forgan

; Assignment: HW1 Assembly Lang Program

; Due Date: 4/2025

; Description: Push 10 4-digit numbers onto the stack, pop the stack 10 times, loop a number of M times

function

M long 7 ; Arbitrary for looping 7 times

sp long 10 ; stack pointer

Start: mov R2, 10 ; put desired stack size in R2

SystemCall 4 ; allocate memory of size in R2 (10)

mov sp, R1 ; place address for memory in sp

add sp, R2 ; move sp to proper location by adding size

mov (sp), 2482 ; push 2482 onto stack

sub sp, 1 ; decrement sp by 1

mov (sp), 4824 ; push 4824 onto stack

sub sp, ; decrement sp by 1

mov (sp), 8242 ; push 8242 onto stack

sub sp, 1 ; decrement sp by 1

mov (sp), 2842 ; push 2842 onto stack

sub sp, 1 ; decrement sp by 1

mov (sp), 3575 ; push 3575 onto stack

sub sp, 1 ; decrement sp by 1

mov (sp), 4567 ; push 4567 onto stack

sub sp, 1 ; decrement sp by 1

mov (sp), 2343 ; push 2343 onto stack

sub sp, 1 ; decrement sp by 1

mov (sp), 9876 ; push 9876 onto stack

sub sp, 1 ; decrement sp by 1

mov (sp), 5303 ; push 5303 onto stack

sub sp, 1 ; decrement sp by 1

mov (sp), 5009 ; push 5009 onto stack

sub sp, 1 ; decrement sp by 1

mov (sp), R1 ; mov value in top to stack to R1

add sp, 1 ; mov sp forward

mov (sp), R1 ; mov value in top of stack to R1

add sp, 1 ; mov sp forward

mov (sp), R1 ; mov value in top to stack to R1

add sp, 1 ; mov sp forward

mov (sp), R1 ; mov value in top of stack to R1

add sp, 1 ; mov sp forward

mov (sp), R1 ; mov value in top to stack to R1

add sp, 1 ; mov sp forward

mov (sp), R1 ; mov value in top of stack to R1

add sp, 1 ; mov sp forward

mov (sp), R1 ; mov value in top to stack to R1

add sp, 1 ; mov sp forward

mov (sp), R1 ; mov value in top of stack to R1

add sp, 1 ; mov sp forward

mov (sp), R1 ; mov value in top to stack to R1

add sp, 1 ; mov sp forward

mov (sp), R1 ; mov value in top of stack to R1

add sp, 1 ; mov sp forward

BrOnPlus M, Start ; BRANCH if M > 0 to start

sub sp, R2 ; put sp to first location of allocated memory

mov R1, sp ; put address of memory in R1

mov R2, 10 ; size of allocated memory into R2

SystemCall 5 ; free the memory

Stop: Halt ; stop - Halt

END Start ; END

Original assembly code for FILE 5:

M long 7 ; Arbitrary for looping 7 times

Start: push 4824 ; push 4 digit number onto stack

push 8242 ; push 4 digit number onto stack

push 2842

push 3575

push 4567

push 2343

push 9876

push 5303

push 5009

pop sp

pop sp

pop sp

pop sp

pop sp

pop sp

pop sp

pop sp

pop sp

pop sp

BrOnPlus M, Start ; BRANCH if M > 0 to start

Stop: Halt ; stop - Halt

END Start ; END

MACHINE LANGUAGE FILE 5:

Push 10 4 digit numbers onto a stack, pop them off, and loop that an arbitrary M times

190 7 // M for M loop iterations

191 106000 // Start: push number onto stack

192 2482

193 106000 // push number onto stack

194 4824

195 106000 // push number onto stack

196 8242

197 106000 // push number onto stack

198 2842

199 106000 // push number onto stack

200 3575

201 106000 // push number onto stack

202 4567

203 106000 // push number onto stack

204 2343

205 106000 // push number onto stack

206 9876

207 106000 // push number onto tack

208 5303

209 106000 // push number onto stack

210 5009

211 110000 // pop number from stack

212 110000 // pop

213 110000 // pop

214 110000 // pop

215 110000 // pop

216 110000 // pop

217 110000 // pop

218 110000 // pop

219 110000 // pop

220 110000 // pop

221 25060 // subtract 1 from M

222 190 // M

223 1

224 85050 // BRANCH if M > 0

225 190 // M

226 191 // Start

227 00000 // stop - HALT

-1 191 // END

Symbol Table (which does not include sp because it was not needed in machine program):

|  |  |
| --- | --- |
| LABEL | ADDRESS |
| M | 190 |
| Start | 191 |
| Stop | 227 |

ASSEMBLY FILE 6:

; Author: Lachlan Forgan

; Assignment: HW1 Assembly Lang Program

; Due Date: 4/2025

; Description: System call to input a character 4 times, then output character 4 times

function

Counter long 4 ; Arbitrary for looping 7 times

Addr long 0

Ptr long 0

Start: mov R2, 9 ; mov 9 into R2 to free 9 words

mov Addr, R1 ; mov address for memory into addr

mov R4, Addr ; copy addr into r4 for iteration

Loop: SystemCall 8 ; input 1st char

mov [R4], R1 ; move char in R1 to address in R4

add R4, 1 ; increment R4

sub Counter, 1 ; decrement counter

BrOnPlus Counter, Loop ; Branch if Counter > 0, go to loop

mov Counter, 4 ; put counter back at 4

mov R4, Addr ; put R4 back at start of allocated memory

Loop2: mov R1, [R4] ; move value in R4 into R1

SystemCall 9 ; output operation, display R1

Sub Counter, 1 ; subtract 1 from counter

Add R4, 1 ; move R4 to next word

BrOnPlus Counter, Loop2 ; if Counter > 0, go to Loop2 (4 iterations)

Mov R1, Addr ; mov Addr into R1

Mov R2, 9 ; move size for memory free (9) into R2

SystemCall 5 ; FreeMemSystemCall

Stop: Halt ; Stop – halt

END Start ; end of program, has value for start

MACHINE LANGUAGE FILE 6:

90 4 // Counter

91 0 // Addr: WORD ADDRESS

92 0 // ptr

93 51260 // Start - move 9 into R2

94 9

95 120000 // System call: free 9 words

96 4

97 55011 // Move start address from R1 into Addr

98 91

99 51450 // copy Addr into R4

100 91

101 120000 // LOOP - system call

102 8 // get 1st char

103 52411 // move char in R1 to address in R4

104 11460 // increase R4 by 1

105 1

106 25060 // Decrease counter by 1

107 90

108 1

109 85050 // BRANCH if counter > 0, go to LOOP

110 90 // LOOP

111 101

112 55060 // Reset 4 into counter

113 90

114 4

115 51450 // Move addr into R4

116 91 // Addr

117 51124 // LOOP2 - Move value in address in R4 into R1

118 120000 // System call

119 9 // io\_putc - display value in R1

120 25060 // Decrement counter by 1

121 90

122 1

123 11460 // Increment R4 by 1

124 1

125 85050 // BRANCH if counter > 0 to LOOP2

126 90 // Counter

127 117 // LOOP2

128 51150 // Mov Addr into R1

129 91 // Addr

130 51260 // Mov size (9) into R2

131 9

132 120000 // System call

133 5 // MemFreeSystemCall

134 000000 // Stop - halt

-1 93 // END

Symbol Table:

|  |  |
| --- | --- |
| LABEL | ADDRESS |
| Coounter | 90 |
| Addr | 91 |
| Ptr | 92 |
| Start | 93 |
| LOOP | 101 |
| LOOP2 | 117 |
| Stop | 134 |

ASSEMBLY LANGUAGE NULL FILE:

function

Start: Branch Start ; branch to start

END Start ; end of program, address for start

MACHINE LANGUAGE NULL FILE:

0 60000 // Loop - branch

1 0 // go to Loop (address 0)

-1 0 // 0 address of start of program