|  |  |  |
| --- | --- | --- |
| cf**Arrow  Description automatically generated** |  | **A picture containing logo  Description automatically generated** |
| **Benha University** |  | **Faculty of Computers & Artificial Intelligence** |

**BBqM™**

**In**

Computer Architecture

**by**

Student’s Full Name (Student Section Number)

|  |
| --- |
| Manar Hamada Elsayed (Section 22) |
| Mona Mohamed Hamdy (Section 22) |
| Menna Muhammad Ibrahim (Section 22) |

**Supervised by**

Eng. Abdallah El Ghamry

Table of Contents

[1. Introduction 1](#_Toc39109705)

[2. Content 2](#_Toc39109706)

[2.1 Inputs and Outputs](#_Toc39109707) 2

[2.2 C-code 2](#_Toc39109708)

2.3  [Flow Chart 6](#_Toc39109708)

2.4  [MIPS Assembly instructions 13](#_Toc39109708)

2.4  [Verify the Operation 14](#_Toc39109708)

2.4  [2 TestCase 14](#_Toc39109708)

# Introduction

In this project, you are going to model the operation of BBqMTM in C Language. Then translate some parts to MIPS assembly language and verify it via simulation utilizing Single-Cycle MIPS processor.

Graphical user interface

Description automatically generated with medium confidence

# Content

## *Inputs and Output*

1. In a table, identify *BBqM™* inputs and outputs and briefly describe their meaning and possible values.

Table ‎2.1 inputs and outputs

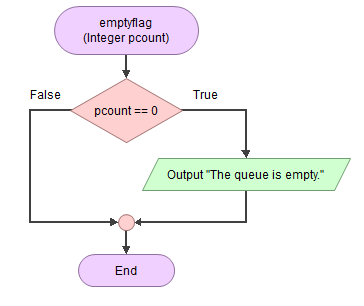
|  |  |  |
| --- | --- | --- |
| **Pin/s Name** | **Type** | **Description** |
| **Reset** | Input | Reset the system |
| **CountUp/Down** | Input | generate “1” if someone enter the queue and generate “0” if someone quit the queue |
| **TCount** | Input | To indicate number of Tellers currently n service {1,2,3} |
| **Pcount** | Output | To indicate number of current people standing in the queue |
| **Wtime** | Output | To indicate Waiting Time |
| **EmptyFlag** | Output | To indicate that queue is empty |
| **FullFlag** | Output | To indicate that queue is full |
| **Alarm** | Output | To indicate that queue is getting full or empty if one interrupt the front photocell or the back photocell |

## *C-code*

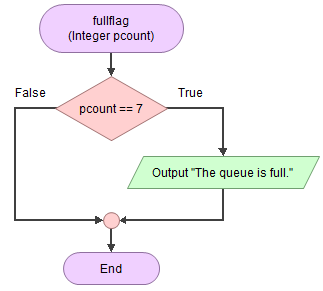
1. #include <stdio.h>
2. #include <stdbool.h>
3. int pcounter (bool updown, int pcount);
4. int tcounter (bool updown, int tcount);
5. void emptyflag (int pcount);
6. void fullflag (int pcount);
7. void p\_alarmflag (int pcount);
8. void t\_alarmflag (int tcount);
9. int waiting\_time (int pcount, int tcount);
10. int main ()
11. {
12. // inputs
13. bool reset;
14. int pcount = 0, tcount = 0, exit, check;
15. // output
16. bool full, empty, alarm;
17. int wtime = 0;
18. // for checking the limits of inputs
19. int tcount\_tst, pcount\_tst;
20. // This statement to welcome the user in first input
21. printf("Welcome to our bank system\n");
22. printf("To add teller enter 3\n");
23. printf("To remove Teller enter -3\n");
24. printf("To count up enter 7\n");
25. printf("To count down enter -7\n");
26. printf("To reset the system enter 0\n");
27. printf("To exit from the system enter -1\n");
28. while (true)
29. {
30. scanf("%d", &check);
31. switch (check)
32. {ذ
33. // if the input is 3 the teller count is increased by 1
34. // with the condition for the alarm if the user crossed the limits
35. case 3:
36. tcount\_tst = tcounter(1, tcount);
37. tcount = tcount\_tst != 4 ? tcount\_tst : tcount;
38. break;
39. // if the input is -3 the teller count is decreased by 1
40. // with the condition for the alarm if the user crossed the limits
41. case -3:
42. tcount\_tst = tcounter(0, tcount);
43. tcount = tcount\_tst != -1 ? tcount\_tst : tcount;
44. break;
45. // if the input is 7 the people count is increased by 1
46. // with the condition for the alarm if the user crossed the limits
47. case 7:
48. pcount\_tst = pcounter(1, pcount);
49. pcount = pcount\_tst != 8 ? pcount\_tst : pcount;
50. break;
51. // if the input is -7 the people count is decreased by 1
52. // with the condition for the alarm if the user crossed the limits
53. case -7:
54. pcount\_tst = pcounter(0, pcount);
55. pcount = pcount\_tst != -1 ? pcount\_tst : pcount;
56. break;
57. // if the input is 0 the program is reset
58. case 0:
59. pcount = 0;
60. tcount = 0;
61. break;
62. // if the input is -1 the program close
63. case -1:
64. exit = -1;
65. break;
66. default:
67. printf("Enter a number from the list above\n");
68. }
69. if (exit == -1)
70. {
71. break;
72. }
73. // Check limits
74. p\_alarmflag (pcount\_tst);
75. t\_alarmflag (tcount\_tst);
76. // check if empty
77. emptyflag (pcount);
78. // Check if full
79. fullflag (pcount);
80. // Print the number of people and waiting time in every order
81. wtime = waiting\_time (pcount, tcount);
82. printf("Waiting time is %d\n", wtime);
83. printf("The Pcount is %d\n\n", pcount);
84. }
85. // This statement to end the program
86. printf("I hope you enjoyed using our small bank system :)\n");
87. return 0;
88. }
89. // The counter of people with limits [0:7]
90. int pcounter (bool updown, int pcount)
91. {
92. if (updown)
93. {
94. if (pcount == 7)
95. {
96. return 8;
97. }
98. pcount++;
99. }
100. else
101. {
102. if (pcount == 0)
103. {
104. return -1;
105. }
106. pcount--;
107. }
108. return pcount;
109. }
110. // The counter of tellers with limits [0:3]
111. int tcounter (bool updown, int tcount)
112. {
113. if (updown)
114. {
115. if (tcount == 3)
116. {
117. return 4;
118. }
119. tcount++;
120. }
121. else
122. {
123. if (tcount == 0)
124. {
125. return -1;
126. }
127. tcount--;
128. }
129. return tcount;
130. }
131. // If empty
132. void emptyflag (int pcount)
133. {
134. if (pcount == 0)
135. {
136. printf("The queue is empty.\n");
137. }
138. }
139. // If full
140. void fullflag (int pcount)
141. {
142. if (pcount == 7)
143. {
144. printf("The queue is full.\n");
145. }
146. }
147. // If crossed the limits for people
148. void p\_alarmflag (int pcount)
149. {
150. if (pcount == 8)
151. {
152. printf("The maximum number of people we can add is 7, Now you only can remove.\n");
153. }
154. else if (pcount == -1)
155. {
156. printf("There is no one in the queue to leave.\n");
157. }
158. }
159. // If crossed the limits for tellers
160. void t\_alarmflag (int tcount)
161. {
162. if (tcount == 4)
163. {
164. printf("The maximun number of tellers is 3, Now you only can remove.\n");
165. }
166. else if (tcount == -1)
167. {
168. printf("There is no tellers to remove.\n");
169. }
170. }
171. // function for calculating the waiting time
172. int waiting\_time (int pcount, int tcount)
173. {
174. int sub, sum, mult;
175. // wtime = 3 \* (pcount + tcount - 1) / tcount
176. // if pcount = 0, wtime = 0
177. if (pcount == 0)
178. {
179. return 0;
180. }
181. // sub = tcount - 1
182. sub = tcount - 1;
183. // sum = sub + pcount
184. sum = sub + pcount;
185. // mult = 3 \* sum // in for loop for assembly
186. mult = 0;
187. for (int i = 0; i < 3; i++)
188. {
189. mult += sum;
190. }
191. // result = mult / tcount
192. int quot = 0;
193. while (mult >= tcount)
194. {
195. mult -= tcount;
196. quot++;
197. }
198. return quot;
199. }

## *2.3 Flow Chart*

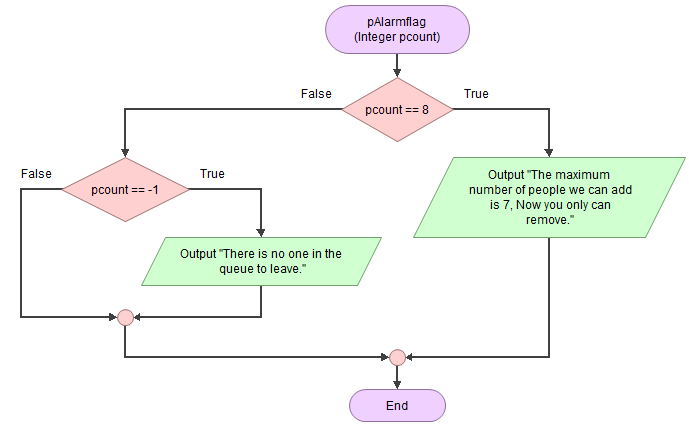
1. Main Function
2. Empty Flag Function

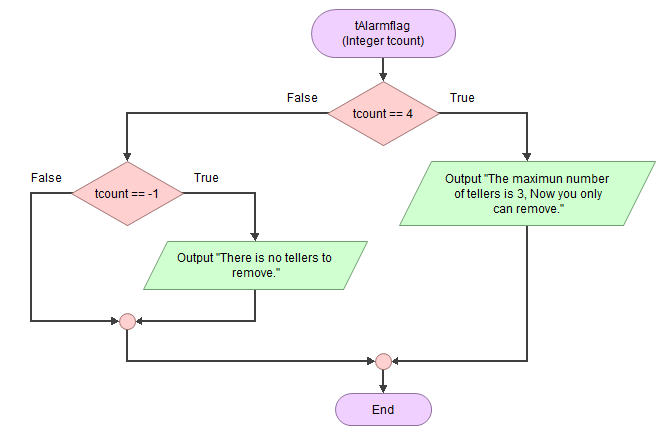
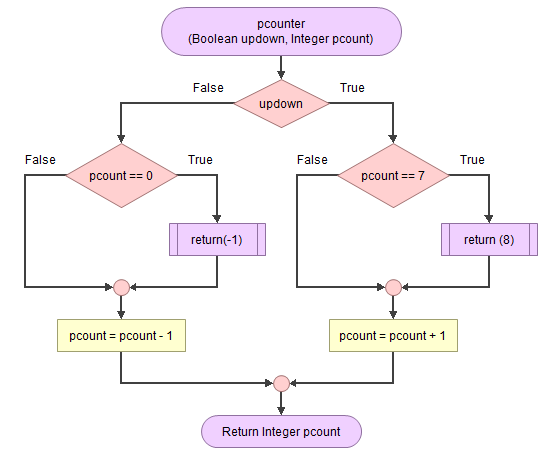
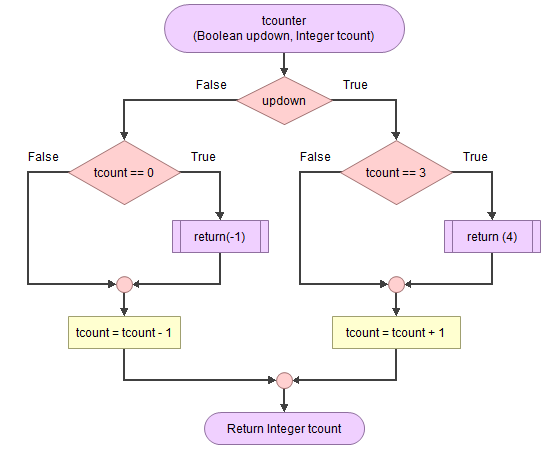


1. Full Flag Function



1. Palarm Flag

// If crossed the limits for people

1. Talarm Flag
2. Pcounter
3. Tcounter
4. Diagram

   Description automatically generatedWaiting Time Function

## *2.4 MIPS Assembly instructions*

|  |  |
| --- | --- |
| Pcount,tcount | $a0,$a1 |
| wtime | $a2 |
| Sub,sum | $t1,$t2 |
| mult | $t3 |
| i | $t4 |
| limit | $t5 |
| quot | T7 |

|  |  |  |
| --- | --- | --- |
| **Assembly** | **Description** | **Machine** |
| **Waiting\_time:** |  |  |
| addi $a0, $zero, 1 | # Pcount = 1 | 20040001 |
| addi $a1, $zero, 1 | # Tcount = 1 | 20050001 |
| addi $a2, $zero, 0 | # Wtime = 0 | 20060000 |
| beq $a0, $zero, End | # If pcount == 0, go to End | 10800016 |
| beq $a1, $zero, End | # If tcount == 0, go to End | 10a00015 |
| addi $t1, $a1, -1 | # Sub = tcount - 1 | 20a9ffff |
| add $t2, $t1, $a0 | # Sum = Sub + pcount | 01245020 |
| addi $t3, $zero, 0 | # mult = 0 | 200b0000 |
| addi $t4, $zero, 0 | # i = 0 | 200c0000 |
| addi $t5, $zero, 3 | # limit = 3 | 200d0003 |
| **For:** |  |  |
| slt $t6, $t4, $t5 | # if i < limit, set $t6 = 1 | 018d702a |
| beq $t6, $zero, Divide | # if $t6 == 0, divide | 11c00005 |
| add $t3, $t3, $t2 | # mult += sum | 016a5820 |
| addi $t4, $t4, 1 | # i++ | 218c0001 |
| J for | # repeat the for loop | 08000c0a |
| **Equal:** |  |  |
| addi $t6, $zero, 1 |  | 200e0001 |
| J continue |  | 08000014 |
| **Divide:** |  |  |
| addi $t7, $zero, 0 | # quot = 0 | 200f0000 |
| **While:** |  |  |
| beq $t3, $a1, Equal | # if mult == tcount | 1165fffc |
| slt $t6, $a1, $t3 | # if mult < tcount | 00ab702a |
| Continue |  |  |
| beq $t6, $zero, Done | # if $t6 == 0, so end the condition | 11c00003 |
| sub $t3, $t3, $a1 | # mult -= tcount | 01655822 |
| addi $t7, $t7, 1 | # quot++ | 21ef0001 |
| J while | #repeat | 08000012 |
| **Done:** |  |  |
| add $a2, $a2, $t7 | # wtime = quot | 00cf3020 |
| sw $a2, 84($zero) | # End, wtime = 0 | ac060054 |
| **End:** |  |  |
| sw $a2, 84($zero) | # End, wtime = 0 | ac060054 |

## *Verify the Operation*

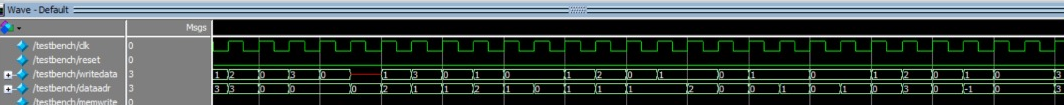
|  |  |  |
| --- | --- | --- |
| T0 | PCOUNT | WTime |
| 1 | 0 | 0 |
| 1 | 1 | 3 |
| 1 | 2 | 6 |
| 1 | 3 | 9 |
| 1 | 4 | 12 |
| 1 | 5 | 15 |
| 1 | 6 | 18 |
| 1 | 7 | 21 |
| 2 | 0 | 0 |
| 2 | 1 | 3 |
| 2 | 2 | 4 |
| 2 | 3 | 6 |
| 2 | 4 | 7 |
| 2 | 5 | 9 |
| 2 | 6 | 10 |
| 2 | 7 | 12 |
| 3 | 0 | 0 |
| 3 | 1 | 3 |
| 3 | 2 | 4 |
| 3 | 3 | 5 |
| 3 | 4 | 6 |
| 3 | 5 | 7 |
| 3 | 6 | 8 |
| 3 | 7 | 9 |

In a table, propose a test strategy to verify the operation of the BBqM™ model. Carefully select an appropriate set of test cases that test various design aspects.

## *2.6 2TestCase*

1. Graphical user interface, application

   Description automatically generatedwhen pcount = 6 & tcount = 2
2. When pcount = 1 & tcount = 1



|  |  |  |  |
| --- | --- | --- | --- |
| **Test\_Case 1 (6,2)** | | **Test\_Case 2**  **(1,1)** | |
| **Address** | **Machine\_code** | **Address** | **Machine\_code** |
| 0x00000000 | 0x20040006 | 0x00000000 | 0x20040001 |
| 0x00000004 | 0x20050002 | 0x00000004 | 0x20050001 |
| 0x00000008 | 0x20060000 | 0x00000008 | 0x20060000 |
| 0x0000000c | 0x10800016 | 0x0000000c | 0x10800016 |
| 0x00000010 | 0x10a00015 | 0x00000010 | 0x10a00015 |
| 0x00000014 | 0x20a9ffff | 0x00000014 | 0x20a9ffff |
| 0x00000018 | 0x01245020 | 0x00000018 | 0x01245020 |
| 0x0000001c | 0x200b0000 | 0x0000001c | 0x200b0000 |
| 0x00000020 | 0x200c0000 | 0x00000020 | 0x200c0000 |
| 0x00000024 | 0x200d0003 | 0x00000024 | 0x200d0003 |
| 0x00000028 | 0x018d702a | 0x00000028 | 0x018d702a |
| 0x0000002c | 0x11c00005 | 0x0000002c | 0x11c00005 |
| 0x00000030 | 0x016a5820 | 0x00000030 | 0x016a5820 |
| 0x00000034 | 0x218c0001 | 0x00000034 | 0x218c0001 |
| 0x00000038 | 0x0800000a | 0x00000038 | 0x0800000a |
| 0x0000003c | 0x200e0001 | 0x0000003c | 0x200e0001 |
| 0x00000040 | 0x08000014 | 0x00000040 | 0x08000014 |
| 0x00000044 | 0x200f0000 | 0x00000044 | 0x200f0000 |
| 0x00000048 | 0x1165fffc | 0x00000048 | 0x1165fffc |
| 0x0000004c | 0x00ab702a | 0x0000004c | 0x00ab702a |
| 0x00000050 | 0x11c00003 | 0x00000050 | 0x11c00003 |
| 0x00000054 | 0x01655822 | 0x00000054 | 0x01655822 |
| 0x00000058 | 0x21ef0001 | 0x00000058 | 0x21ef0001 |
| 0x0000005c | 0x08000012 | 0x0000005c | 0x08000012 |
| 0x00000060 | 0x00cf3020 | 0x00000060 | 0x00cf3020 |
| 0x00000064 | 0xac060054 | 0x00000064 | 0xac060054 |
| 0x00000068 | 0xac060054 | 0x00000068 | 0xac060054 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test\_Case 2** | | | **Test\_Case 1** | | |
| **Time** | **Data\_address** | **Write\_data** | **Time** | **Data\_address** | **Write\_data** |
| 22 | 6 | 6 | 22 | 1 | 1 |
| 30 | 2 | X | 30 | 1 | X |
| 40 | 0 | X | 40 | 0 | X |
| 50 | 6 | 0 | 50 | 1 | 0 |
| 60 | 2 | 0 | 70 | 0 | X |
| 70 | 1 | X | 80 | 1 | 1 |
| 80 | 7 | 6 | 90 | 0 | X |
| 90 | 0 | X | 110 | 3 | X |
| 110 | 3 | X | 120 | 1 | 3 |
| 120 | 1 | 3 | 130 | 1 | 0 |
| 130 | 1 | 0 | 140 | 1 | 1 |
| 140 | 7 | 7 | 150 | 1 | 0 |
| 150 | 1 | 0 | 160 | 0 | 0 |
| 160 | 0 | 0 | 170 | 1 | 3 |
| 170 | 1 | 3 | 180 | 1 | 0 |
| 180 | 1 | 0 | 190 | 2 | 1 |
| 190 | 14 | 7 | 210 | 0 | 0 |
| 200 | 2 | 1 | 220 | 1 | 3 |
| 210 | 0 | 0 | 230 | 1 | 0 |
| 220 | 1 | 3 | 240 | 3 | 1 |
| 230 | 1 | 0 | 250 | 3 | 2 |
| 240 | 21 | 7 | 260 | 0 | 0 |
| 250 | 3 | 2 | 270 | 0 | 3 |
| 260 | 0 | 0 | 280 | 0 | 0 |
| 270 | 0 | 3 | 290 | 0 | X |
| 280 | 0 | 0 | 300 | 2 | 1 |
| 290 | 0 | X | 310 | 1 | 3 |
| 300 | 19 | 2 | 320 | 1 | 0 |
| 310 | 1 | 21 | 330 | 2 | 1 |
| 320 | 1 | 0 | 340 | 1 | 0 |
| 330 | 19 | 2 | 350 | 0 | 0 |
| 340 | 1 | 0 | 360 | 1 | 1 |
| 350 | 0 | 0 | 370 | 1 | 2 |
| 360 | 17 | 2 | 380 | 1 | 0 |
| 370 | 1 | 19 | 390 | 1 | 1 |
| 380 | 1 | 0 | 400 | 2 | 1 |
| 390 | 17 | 2 | 410 | 0 | 0 |
| 400 | 2 | 1 | 420 | 0 | 1 |
| 410 | 0 | 0 | 430 | 1 | 1 |
| 420 | 15 | 2 | 440 | 0 | 0 |
| 430 | 1 | 17 | 450 | 1 | 0 |
| 440 | 1 | 0 | 460 | 0 | 1 |
| 450 | 15 | 2 | 470 | 3 | 2 |
| 460 | 3 | 2 | 480 | 0 | 0 |
| 470 | 0 | 0 | 490 | 4294967295 | 1 |
| 480 | 13 | 2 | 500 | 0 | 0 |
| 490 | 1 | 15 | 520 | 3 | 3 |
| 500 | 1 | 0 | 530 | 84 | 3 |
| 510 | 13 | 2 |
| 520 | 4 | 3 |
| 530 | 0 | 0 |
| 540 | 11 | 2 |
| 550 | 1 | 13 |
| 560 | 1 | 0 |
| 570 | 11 | 2 |
| 580 | 5 | 4 |
| 590 | 0 | 0 |
| 600 | 9 | 2 |
| 610 | 1 | 4 |
| 620 | 1 | 0 |
| 630 | 9 | 2 |
| 640 | 6 | 11 |
| 650 | 0 | 0 |
| 660 | 7 | 2 |
| 670 | 1 | 5 |
| 680 | 1 | 0 |
| 690 | 7 | 2 |
| 700 | 7 | 9 |
| 710 | 0 | 0 |
| 720 | 5 | 2 |
| 730 | 1 | 7 |
| 740 | 1 | 0 |
| 750 | 5 | 2 |
| 760 | 8 | 7 |
| 770 | 0 | 0 |
| 780 | 3 | 2 |
| 790 | 1 | 5 |
| 800 | 1 | 0 |
| 810 | 3 | 2 |
| 820 | 9 | 8 |
| 830 | 0 | 0 |
| 840 | 1 | 2 |
| 850 | 1 | 3 |
| 860 | 1 | 0 |
| 870 | 1 | 2 |
| 880 | 10 | 9 |
| 890 | 0 | 0 |
| 900 | 4294967295 | 2 |
| 910 | 0 | 1 |
| 920 | 0 | 0 |
| 930 | 10 | 10 |
| 940 | 84 | 10 |