|  |
| --- |
| Part a:  Part b:  Part c:      Part a:  Part b: |
| |  |  |  | | --- | --- | --- | | 4008 | 1100101 | e | | 4007 | 1101110 | n | | 4006 | 1100001 | a | | 4005 | 1101100 | l | | 4004 | 1110000 | p | | 4003 | 1101111 | o | | 4002 | 1110010 | r | | 4001 | 1100101 | e | | 4000 | 1000001 | A |   I did parts 1 & 2 in an Excel spreadsheet and then screenshotted the data for convenience. I hope this is okay. |
| // Problem 4  // converting MIPS to C  // Parker Hague  int x = y; // add function: adding two elements and storing in a variable  int a;  // slt function: compares two values and stores a 1 if y < z or a 0 if false  // beq function: checks if x == 0  if (y < z && y == 0){  a = 1;  int b = c - x;  }  else{  a = 0; // false for beq function  x = z; // executes if beq function is false  } |
| // problem 5  // MIPS to C  // Parker Hague  int a = 0; // add 0 + 0  int b = 0; // add 0 + 0  int c = 20; // add 0 + 20  int d; // $t1  // combined slt & beq functions  // jump with condition creates while loop  while (b < c && b != d){ // have to use != because if beq func is true then the system will exit  d = 1;  if (b == d){  break; // exit if true  }  else{  d = 0; // false of slt function  a = a + b; // false of beq function  }  b++; // b = b + 1  } |
| # Problem 6  # C to MIPS  # Parker Hague  add $t0, $zero, $zero # $t0 = $zero + $zero  sll $t1, $s1, 2 # i \* 4 ... ith index of array  sll $t2, $s2, 2 # j \* 4 ... jth index of array  add $t1, $t1, $s0 # $t1 = $t1 + $s0  add $t2, $t2, $s0 # $t0 = $t1 + $t2  lw $t3, 0($t1) # $t3 = A[i]  lw $t4, 0($t2) # $t4 = A[j]  bne $t3, $t4, True # if $t3 != $t4 then True  j False # jump to False  # if bne is false, True block doesn't get executed    True:  add $t0, $t3, $t3 # $t0 = $t3 + $t4  False: # false statement will execute regardless of bne result  lw $t5, 0($s0)  add $t0, $t0, 0($s0) # $t0 = $t0 + 0($s0) |
| #problem 7  #Parker Hague  add $s1, $zero, $zero # $s1 = $zero + $zero...i variable  addi $t0, $zero, 10 # $t0 = $zero + 10... 10  L1:  slt $t2, $s1, $t1  beq $t2, $zero, Exit # if $t2 == $zero then Exit  sll $t3, $s1, 2 # i \* 4  add $t3, $t3, $s0 # element at A[i]  sw $s1, 0($t3) # stores array index in array location  addi $s1, $s1, 1 # $s1 = $s1 + 1    j L1 # reiterates through for loop    Exit:  add $s1, $zero, $zero # $s1 = $zero + $zero...sets i back to zero  addi $t3, $zero, 5 # $t3 = $zero +5  addi $t5, $zero, 9 # $t5 = $zero + 9  L2:  beq $s1, $t3, Done # checks if i = 5  sll $t3, $s1, 2 # i \* 4 for ith index  add $t3, $t3, $s0 # adds i index to A array creating element  lw $t0, 0($t3) # loads i index into temp  sub $t4, $t5, $s1 # $t4 = 9 - i  sll $t4, $t4, 2 # multiply by 4 for array address  add $t4, $t4, $s0 # adds for array value  lw $t6, 0($t4) # A[9 - i]  sw $t6, 0($t3) # A[i] = A[9 - i]  sw $t0, 0($t4) # A[9 - i] = temp  addi $s1, $s1, 1 # $s1 = $s1 + 1...i += 1  j L2    Done: |
| #Problem 8  # Parker Hague  # $s0 = A[0]  # $s1 = i  # $s2= j  # $s3 = M  # $s4 = N  add $s1, $zero, $zero # $s1 = $zero + $zero i = 0  add $s2, $zero, $zero # $s2 = $zero + $zero  L1:  beq $s1, $s3, Exit # if $s1 == $s3 then Exit  L2:  beq $s2, $s4, End # $s2 == $s4 then break out of loop  add $t0, $s1, $s2 # $t0 = $s1 + $s2  sll $t0, $t0, 2 # i \* j \* 4  add $t0, $t0, $s0 # $t0 = $t0 + $s0  mul $t1 $t1, $s1, $s2 # multiply i \* j and store into $t1  sw $t1, 0($t0) #stores i \* j into A[i + j]  addi $s2, $s2, 1 # $s2 = $s2 + 1  j L2  End:  addi $s1, $s1, 1 # $s1 = $s1 + 1  j L1 # loops L1  Exit: |
| #Parker Hague  #problem 9  # $s0 = temp1  # $s1 = temp2  # $t0 = i  addi $s0, $s0, 9 # $s0 = $s0 + 9 temp1  addi $s1, $s1, 0 # $s1 = $s1 + 0 temp2  add $t0, $zero, $zero # $t0 = $zero + $zero i  addi $t1, $t1, 10 # $t1 = $t1 + 10    slt $t2, $t1, $s0 # if 10 < temp1  beq $t2, 0, Else # executes Else if temp1 < 10  If:  beq $t0, $t1, Break  add $s1, $s1, $t0 # $s1 = $s1 + $t0 # temp2 = temp2 + i  addi $t0, $t0, 1 # $t0 = $t0 + 1 i++  j If    Else:  addi $t3, $zero, -10 # sets t3 to -10  add $t0, $zero, $zero # $t0 = $zero + $zero i  L2:  beq $t0, $t3, Break # if $t0 == $t3  add $s1, $s1, $t0 # $s1 = $s1 + $t0 # temp2 = temp2 + i  addi $t0, $t0, -1 # decrements in for loop  j L2 # loops  Break: |
| #Parker Hague  #Problem10  # A[] = $s0  # x = $t0  # i = $t1  # j = $t2  # h = $t3  L1:  sll $t4, $t1, 2 # i \* 4  add $t4, $t4, $s0 # adds i index to A[]  lw $t5, 0($t4) # loads index into register  add $t0, $t0, $t5 # $t0 = $t0 + $t5 x = x + A[i]  add $t1, $t1, $t2 # $t1 = $t1 + $t2  beq $t1, $t3, Break # if i == j  j L1; # loops back to top  Break: |