**Contents:**

* **Clear screen** subroutine
* **Box subroutine**: prints a rectangle. Takes in five parameters
  + Byte attribute of rectangle
  + Length of rectangle
  + Breadth of rectangle
  + X-position of rectangle
  + Y-position of rectangle
* **Hurdle subroutine**: Will call Box subroutine 3 times. Once for its entire length. Twice for additional space so player can stand. Every hurdle will have equal width of 4 words, and byte attribute of Red. Length, x- position and y-position will be passed in parameters.
* **Mario Subroutine**: prints the character, which is divided into parts such as body, head, legs, hands and neck, each part calling the box subroutine separately (except for head as we would be printing face in head as well). Height, width and attribute of character remains constant, X and Y position will be passed through parameters.
* **Game flag subroutine**: Prints a green flag in top right corner of screen. Also prints a star within the flag.
* **Delay** subroutine: simply delays the program for visual aid of user
* **ClrMario** subroutine: clears the previously printed Mario character to show movement as next Mario is printed on another location. This will prevent clearing the entire screen and printing the background every time with every movement. This subroutine will take parameters just like that of the normal Mario subroutine does.
* **Kbisr** (ISR): main Interrupt service request (hardware interrupt) which will make use of keystrokes in the game.
  + The first comparison will check if right key is pressed and move the character accordingly
  + Else move to “Nextcmp”
  + We will further compare with values if collision with hurdles is happening, if collision, the character will not move and far jump will call original ISR
  + We will also compare if end of hurdle is reached, in case that character is moving on top of the hurdle. In that case, character will fall back on ground
* **Nextcmp [label]**
  + Likewise, we check if left key is pressed and move the character accordingly
  + Else move to “Nextcmp2”
  + We will further compare with values if collision with hurdles is happening, if collision, the character will not move and far jump will call original ISR
  + We will also compare if end of hurdle is reached, in case that character is moving on top of the hurdle. In that case, character will fall back on ground
* **Nextcmp2 [label]**
  + We compare if “UP” key is pressed or not, else move to “Nextcmp3”
  + Normal working of up key would be that the character would jump up 9 lines, and then fall back on ground after a little bit of delay to show the actual representation of “jump”.
  + Otherwise, if character jumps while standing adjacent to any hurdle, will jump on top of the hurdle… as multiple key strokes cannot be detected nor a requirement in the given instructions, standing right next to a hurdle and then pressing “UP” key be enough for character to jump on top of that hurdle and move on top.
  + If the character is on left side of hurdle, it will automatically move once towards right, or if it is on right side of hurdle, it will automatically move once towards left.
  + For abovementioned functionality, the data labels: **rup1, rup2, rup3, lup1, lup2, lup3** have been used.
  + Use of labels: **rup1e, rup2e, rup2e, lup1ee, lup2ee, lup3ee, lup1e, lup2e, lup3e**; have been used as not using these intermediary jumps would lead to the reach of short jump being not enough and cause syntax error.
* **Nextcmp3 [label]**
  + Calls the original ISR through far jump
* **Exit [label]**
  + Sends EOI and exits ISR
* **TSR** end of the code is Terminate and Stay resident.
* **Data labels** data labels used for x and y position of the character are **xpos** and **ypos** respectively. Data label for storing original ISR is **oldisr**
* **PrintScreen1**
  + Prints name of the game and Loading screen. User can press any key to go to the next screen
* **GetName**
  + Gets the name of user and stores it in a label. User name is displayed throughout the game of top left corner.
* **Timer**
  + ISR for timer interrupt. Enemy moves after every 60 clicks
* **Win** 
  + Checks if character has touched the flag. If it has touched the flag, then the game stops and Bios interrupts are used to display player name and “You Win “message.
* **Lose**
  + Checks if character has touched the enemy. If it has touched the enemy, then the game stops and Bios interrupts are used to display player name and “You Lose “message. Note: Enemies and monsters can be added to the code. They are not included in this code.
* **Clrdash**
  + Prints dashes on entire screen with a bit delay to seem like animation
* **Delay2**
  + This has a shorter interval time than subroutine Delay
* **Background**
  + Kingdom is printed in background using Kwrap and Kingdom.