IO

Crafting Terminal:

Inputs:

1. Item Selection Panel
   1. This is likely a storage place full of all the items that are programmed into the system. In the Minecraft version, it will be a chest full of all the programmed items that the CraftingProcessingUnit can craft. The names of those items will be a 10-bit memory address, stored in binary. In the program version, it will just be a button that, when clicked, will change from one to zero and from zero to one.
2. Number of items wanted panel
   1. A place where the user chooses how many items they need from the system. This is imagined as a number pad of buttons that, when pressed, act like a microwave seven-segment display. The maximum amount of items that can be requested from the system will be 9,999, so four seven-segment displays are used. The keypad layout resembles a typical numeric keypad for ease of use.
3. Craft button
   1. A button that tells the system to craft the number of items the user chose to craft.
4. Clear button
   1. This button clears all the information in the inputs and the outputs.

Outputs:

1. How many of the selected item do we already have in storage?
   1. If we have the amount of items the user wants to craft already in storage, then just ship those items to the user. If there is any at all in storage, use them, then craft the remaining items needed to fulfill the user's requests.
2. How many of the selected items are currently able to be crafted, taking into account all dependencies?
   1. If we are crafting a Birch Door and we don’t have any birch wood, then see if we have a birch wood recipe. If so, see if enough birch logs exist to craft the birch wood required to craft the birch door.
   2. If we are crafting a Blast Furnace:
      1. Request a furnace from storage. If one exists, get it; otherwise, make one.
      2. Request 5 smooth stones from storage. If none exist, smelt 5 smooth stones; if none exist, smelt 5 cobblestones.
      3. Request iron from storage. If none exist, look at the recipes for iron known. If we choose to smelt ores, do that; if we choose to craft iron blocks into 9 iron ingots, take three and store the rest in storage.
   3. If the system does not have the ability to craft the number the user wants, the system will not do anything.
3. Crafting?
   1. If the system is currently crafting, it will NOT accept any new jobs. This is just a lamp that is on if the system is busy.
4. Amount remaining to be crafted:
   1. A seven-segment display panel that shows the number of items still needed to be crafted.

Recipe Terminal:

Inputs:

1. Crafting or smelting?
   1. This is a toggle button; off is crafting, on is smelting, and the default is off.
2. If crafting, we need 9 addresses to items in storage.
   1. The UI has 10 bits that act the same way as in the Crafting Terminal. However, this time there is a button that will rotate through each of the inputs and outputs. There will be a seven-segment display that starts up at 1. If we input addresses into the system and click the button, it will count up to 2, and so on. The important thing is that when the button is clicked and the values are all changed back to their default values of zero, it remembers them so that when the button is pressed enough times to roll over from 9 to 1, 1 will have the original values placed inside.
3. If smelting, we need 1 address to the items being smelted.
   1. This uses the same buttons as the previous ones, except when smelting there is only one input item, so we only have to remember the singular address. This means that if the crafting/smelting button is on, the switch address being put in will get disabled, and only the first address will get sent to the greater program later.
4. If crafting, amount of resource made per craft.
   1. This is a spot where the user can change the default number of items that will be crafted per craft. For example, when you make doors out of six oak planks, it makes three. When inputting the crafting recipe, we have to make sure to input 3 in this field so the computer knows how many times to craft an item. Also, the computer will send any extra items to storage if it accidentally makes more than the user wanted. If the user says he wants 2 doors, the last one will get sent to storage. This field uses a numeric keypad with two seven-segment displays, and the maximum value is limited to 64 to match Minecraft's stack size limits. There is also a "Clear Amount" button to reset this field.
5. Address of crafted resource in memory, so that other recipes can use it.
   1. Using the same system for the 10 memory bits, this information will be used to store whatever items you're programming to craft in memory.
6. Submit
   1. A submit button that will take all of this data and send it to the computer.
   2. All of the input data should be cleared after this button is pressed.
7. Clear
   1. This button clears the inputs from all the input fields.

Outputs:

1. Working?
   1. Is the system busy?
2. Current item programming 1-9
   1. This is just the seven-segment display that shows the user what address they are programming at this moment.

CPU - Crafting Processing Unit

Inputs from Crafting Terminal:

1. Memory Address of item to be crafted.
2. Amount to be crafted.
3. Takes a Signal to start Crafting.
   1. This operation will be available after the feasibility of the Job has been computed and sent back to the User Interface.

Outputs to Crafting Terminal:

1. Crafting?
   1. A bit of information that if true the IO will not send any more info to the CPU until it is done with its current task.
2. amount of Item to be crafted in storage.
3. Maximum number of selected items that can be crafted.
4. Items remaining to be crafted in the current craft job.

Inputs from Recipe Terminal:

1. Something
2. Something

Ouputs to Recipe Terminal:

1. Something
2. something

Algorithm:

IDK yet

Memory:

Minecraft 1.20 has 1,643 different items:

12 bit lookup: 2^12 = 4,096 different items storable

10 bit lookup: 2^10 = 1,024 different items storable

8 bit lookup: 2^8 = 256 different items in storage

Memory controller:

I need the memory controller to automatically allocate memory when deleting items, then when adding a Furness recipe to the memory I need one memory location stored while when storing a typical crafting recipe I could have anywhere from one to nine memory locations stored, so I need something to keep track of that.

Memory

My memory will be simple, it will have 10 lookup bits accessible though 2x5 bit pulses, this will mean I will have to do some transcoding to and from 10 bits but it will halve the wiring I need to do.

Most of the memory will be allocated for crafting table recipe storage because at each address there needs to be at most 9x10 bit look ups for each of the nine items in the recipe

There will be a signal that is separate from the data that tells the memory to output data at the specified address or to overwrite memory at that address.

The database will need to take 11 lists of 10 bits.

A screenshot of a computer program

Description automatically generated

The first nine are address bits for memory locations that store item addresses per slot in the crafting table if these are all zeros that means that spot in the recipe is blank.

Next list of 10 bits is the location in memory all this information should be located under, and the location in storage that will store any extra items crafted.

And last is a list of 10 bits that that the first one signifies if the recipe is ment for a crafting table or a furness, if it is a 0 that means the recipe is for a crafting table, if it is a one that means the recipe is for a furness. The next 2 bits are place holders incase I want to add functionality later, and the last 7 are for storing the binary number up to 64 in size.