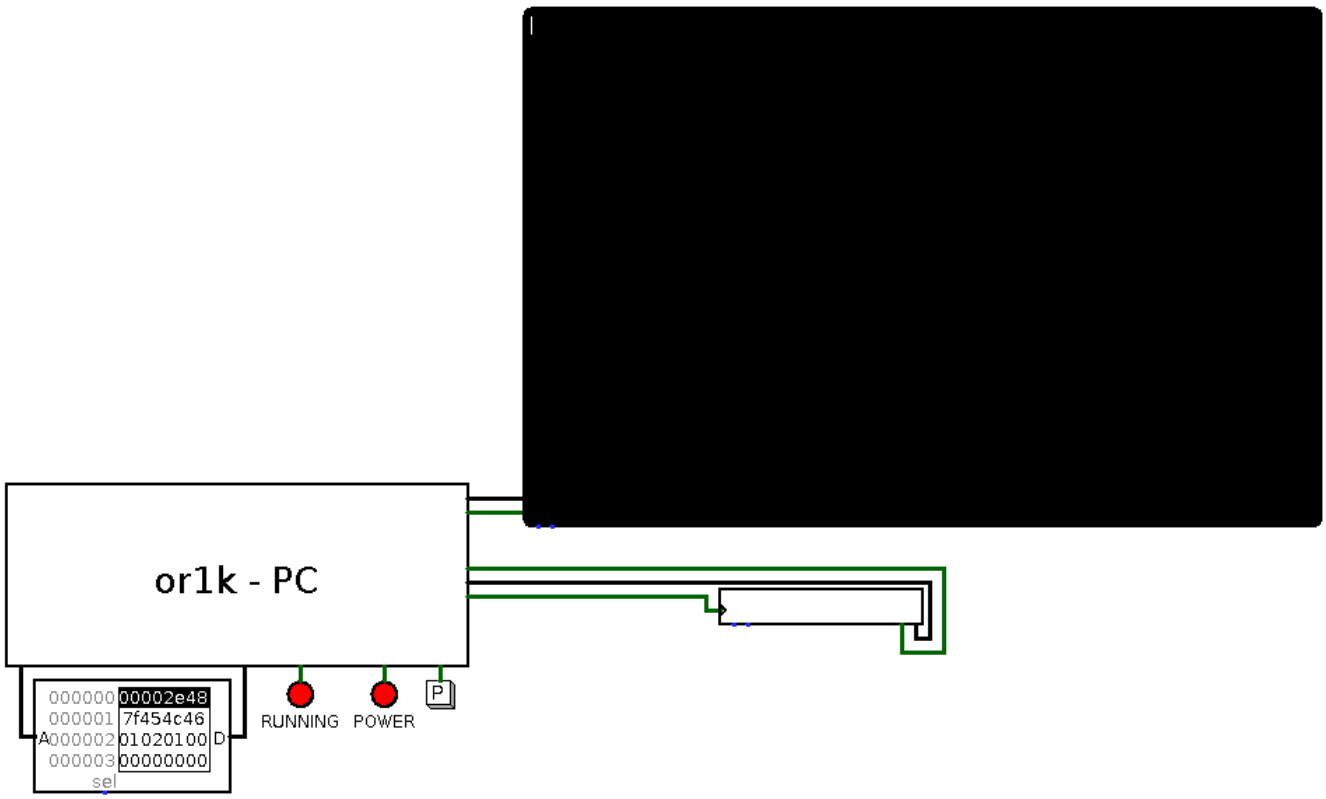
CPU or1k su Logisim

Dario Ostuni



L'obiettivo principale di questo progetto è creare un circuito sul programma Logisim che sia in grado di eseguire programmi precompilati scritti in C/C++.
Le opzioni per arrivare a tale obiettivo erano inventare un istruction set e poi scrivere un backend per tale instruction set su un compilatore come GCC o LLVM oppure implementare un istriction set esistente di cui sia implementato il backend in un qualche compilatore di C/C++.
La scelta è ricaduta sulla seconda opzione, in quanto per la prima ci sarebbe voluto molto più tempo di quanto non è stato usato per realizzare la prima opzione.
L'istruction set utilizzato è stato scelto secondo i seguenti criteri:

- deve essere di tipo RISC;
- deve essere royalty-free e open-source;
- deve esserci un backend di tale architettura per GCC o LLVM.

La scelta dell'architettura si è ristretta quindi a OpenRISC (o or1k) o SPARC. È stato preferito il primo in quanto ha meno istruzioni.
A questo punto quindi i componenti necessari per raggiungere l'obiettivo iniziale sono:

- una CPU or1k per eseguire le istruzioni;
- una RAM per avere una memoria per i programmi che verranno eseguiti;
- una “tastiera” per permettere all'utente di dare input al programma;
- un “terminale” per permettere all'utente di vedere l'output del programma;
- una ROM da cui caricare il programma;
- un “pulsante” con cui attivare il circuito.

Le azioni che quindi dovranno essere eseguite sono:

1. se viene premuto il “pulsante” d'accensione, copia il programma dalla ROM alla RAM;
2. carica l'istruzione attuale del programma in CPU ed eseguila;
3. se c'è qualcosa nel buffer di output, scrivilo sul “terminale”;
4. se c'è qualcosa sulla “tastiera”, scrivilo nel buffer di input;
5. torna al punto 2.

Concetti generali

Il circuito è stato elaborato per essere il più concettualmente semplice possibile, anche se questo vuol dire dare poco peso alle prestazioni risultanti.
I vari componenti sono stati astratti come “host” di un network di tipo Token Ring. Nello specifico, c'è un token che gira per il circuito che indica qual è il componente attualmente attivo, il quale tiene il token fino a quando non ha finito di svolgere la propria operazione e successivamente lo rilascia al componente dopo.
La CPU non implementa al 100% l'architettura OpenRISC, ma un subset sufficiente a far eseguire programmi compilati con GCC. Nello speficico sono stati implementati come da specifica or1k:

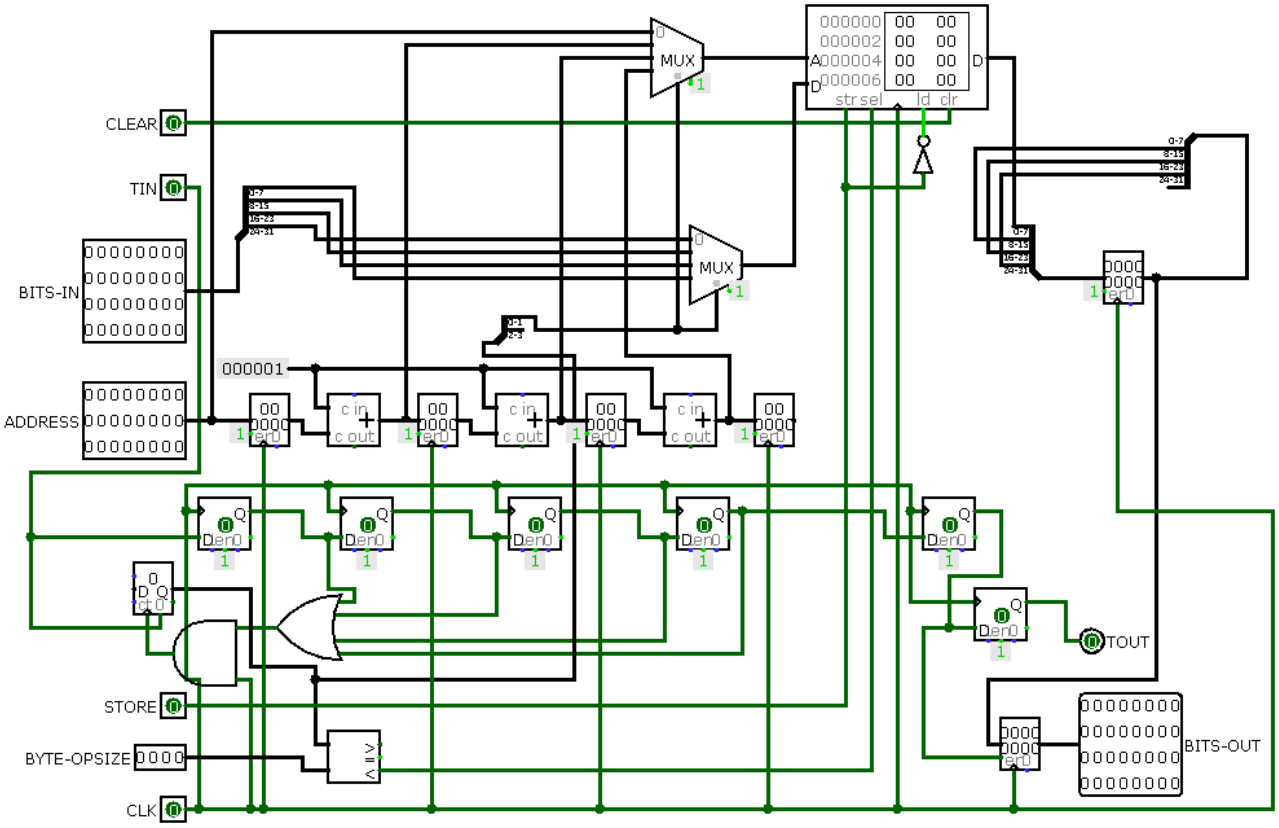
- un Program Counter (PC);
- un Istruction Register (IR);
- 32 registri generici (R1..R32);
- un flag generico (F);
- tutte le istruzioni di jump tranne quelle riguardanti le eccezioni;
- tutte le istruzioni di load;
- tutte le istruzioni di store;
- tutte le istruzioni di comparazione;
- tuttele istruzioni aritmetico-logiche senza la gestione delle eccezioni e le istruzioni di estensione degli interi.

In or1k 4 dei 32 registri generici hanno delle particolari convenzioni:

- il registro 0 deve sempre avere valore 0;
- il registro 1 dovrebbe contenere lo stack pointer;
- il registro 2 dovrebbe contenere il frame pointer;
- il registro 9 dovrebbe contenere l'indirizzo della funzione chiamante.

La CPU terminerà l'esecuzione del programma quando la funzione principale farà un salto all'indirizzo contenuto nel registro r9.

Circuiti

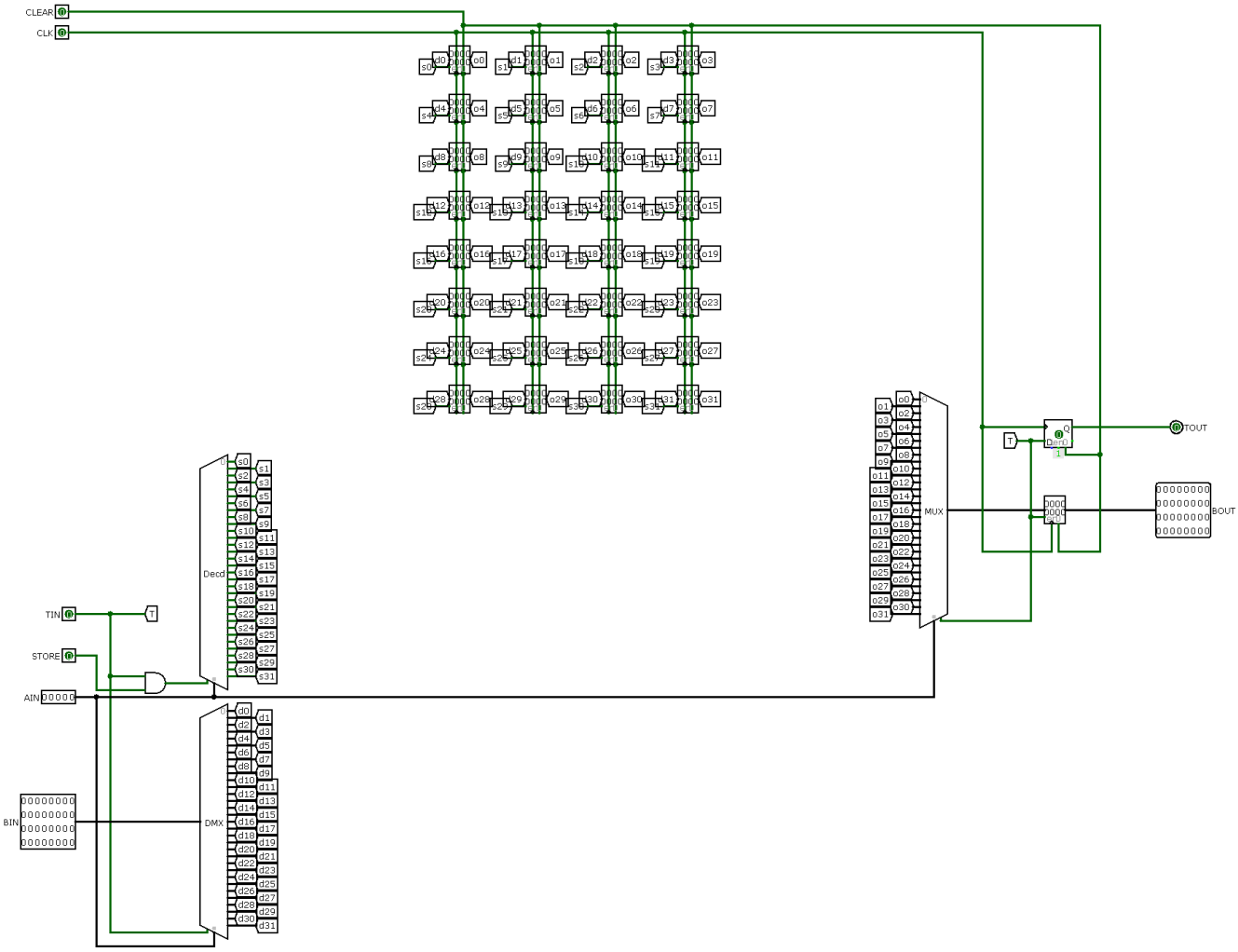


La RAM:
Questo componente rappresenta l'interfaccia alla RAM di Logisim, in quanto il componente in se permette solo di fare letture e scritture di lunghezza statica. Grazie a questa interfaccia viene aggiunto il supporto ai token e letture/scritture a lunghezza variabile.
L'input è formato da:

- CLEAR: quando questo input va sul livello alto la RAM viene resettata;
- TIN: l'ingresso del token;
- BITS-IN: i bit che devono essere scritti in RAM;
- ADDRESS: l'indirizzo in cui devono essere scritti/letti i bit;
- STORE: se il bit è al livello alto allora viene effettuata una scrittura, altrimenti una lettura;
- BYTE-OPSIZE: il numero di byte che deve essere scritto/letto, accetta solo 1, 2 o 4 come ingresso;
- CLK: il segnale di clock.

L'output è formato da:

- TOUT: l'uscita del token;
- BITS-OUT: i bit che sono stati letti.



I registri della CPU:

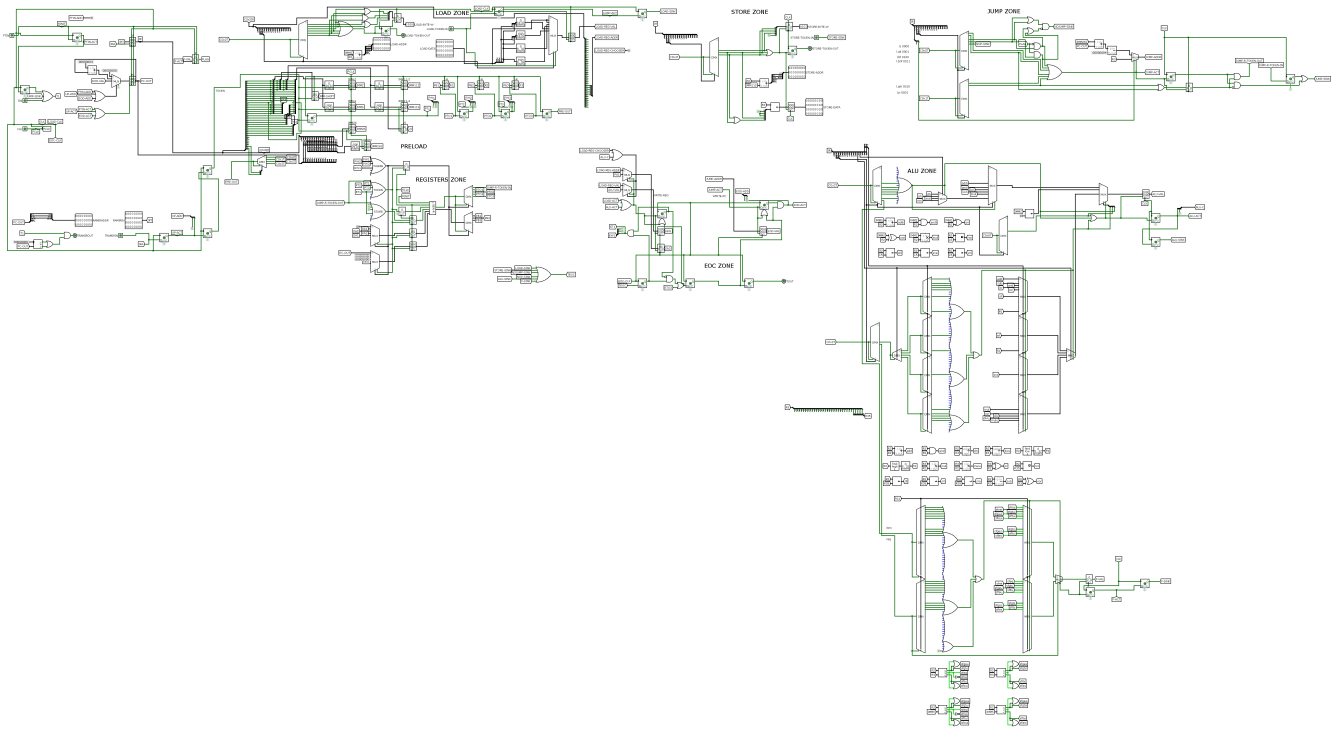
In questo componente vengono gestiti i 32 registri generici della CPU, con l'aggiunta del supporto ai token.

L'input è formato da:

- CLEAR: quando questo input va sul livello alto i registri vengono resettati;
- CLK: l'ingresso del clock;
- TIN: l'ingresso del token;
- STORE: se il bit è al livello basso viene effettuata una lettura, altrimenti una scrittura;
- AIN: numero del registro sul dove deve essere eseguita l'operazione.

- BIN: i dati che devono essere scritti.

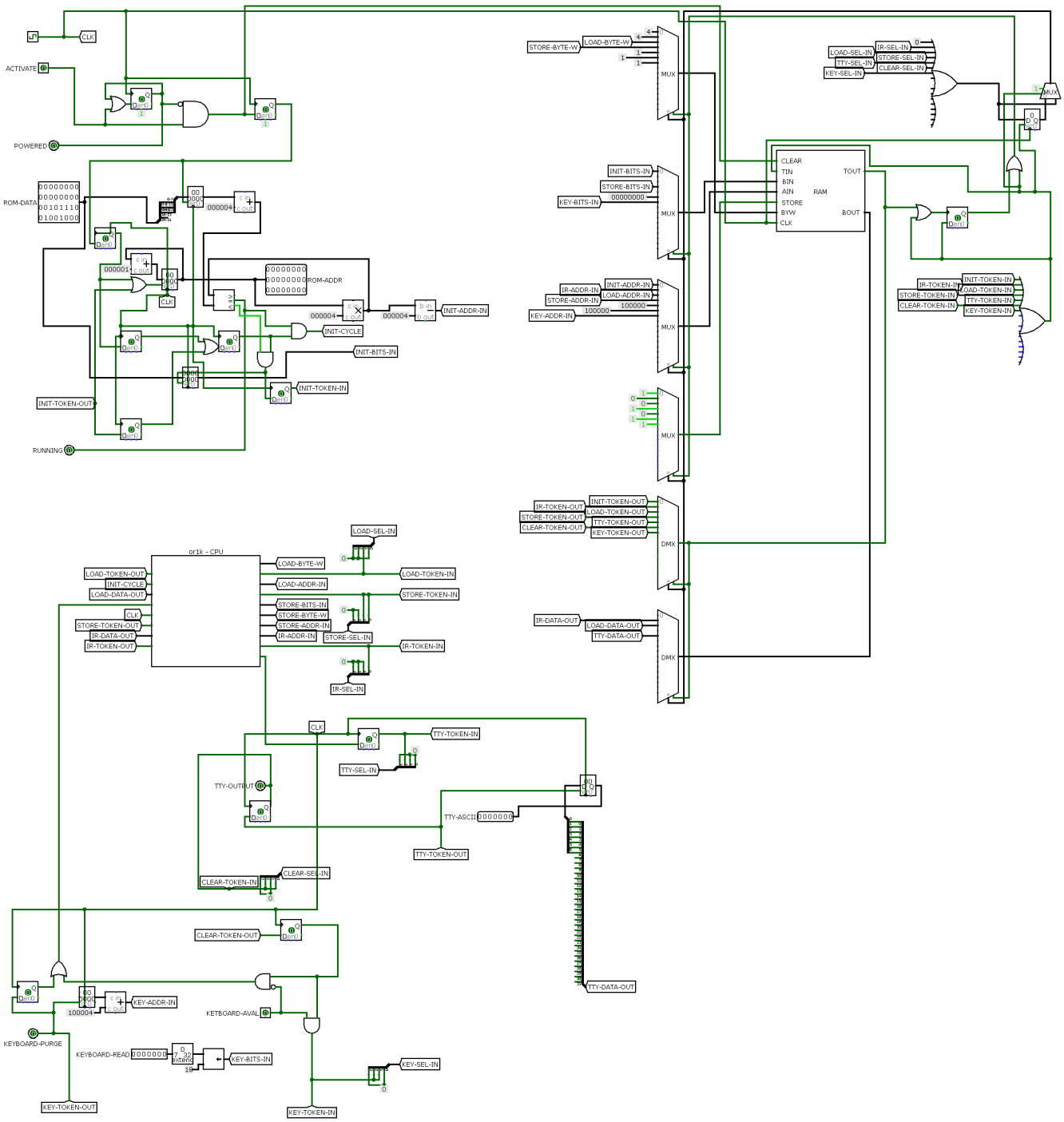
L'output è formato da:
- TOUT: l'uscita del token;
- BOUT: i dati che sono stati letti.



La CPU:
In questo componente vengono eseguite le istruzioni del programma caricato. La prima volta che riceve il token il componente viene inizializzato, nello specifico il Program Counter viene impostato a **0x00002000**, il Flag a **0** e vengono azzerati i registri. Dalla seconda volta in poi in cui viene ricevuto il token vengono eseguite le seguenti operazioni:
- Se il PC ha il valore **0** viene arrestata l'esecuzione del programma;
- viene caricata dalla RAM l'istruzione del programma puntata dal PC e viene messa nell'IR (Istruction Register);
- vengono caricate in dei registri speciali le informazioni dell'istruzione, come l'op-code, i 3 registri che potrebbe contenere l'istruzione (rA, rB, rD) e l'immediato esteso con zeri e con segno;
- viene eseguita l'istruzione, attivando il circuito

adeguato tramite un demultiplexer attivato con l'op-code dell'istruzione. Eventuali salti o scritture nei registri risultanti dall'istruzione verranno scritti in dei registri speciali;

- se è stato eseguito un salto viene eseguita l'istruzione successiva prima di saltare (come da specifica or1k;
- riparte il ciclo della CPU.



Il circuito principale:
Questo circuito serve a unire tutti i componenti, a inizializzare la RAM e gestire l'I/O.
Quando viene avviato prende i contenuti della ROM e li copia in RAM, dopodiché viene generato il token e viene passato alla CPU; una volta restituito vengono eseguite le procedure per gestire la stampa sulla tty e la lettura dalla tastiera.

Bug noti:

- per un bug di GCC le variabili globali nei programmi sono inutilizzabili;
- per un bug di BFD (il linker) la funzione “main” non viene messo come entry point, bensì viene messa come entry point la prima funzione dichiarata nel programma.

Specifiche finali

Lo scopo di questo progetto è creare un circuito che sia in grado di eseguire programmi scritti in C/C++ e compilati con GCC in formato ELF per l'architettura OpenRISC 1000 (or1k).
L'interfaccia utente consiste in 3 dispositivi di input e 3 dispositivi di output:

- un terminale (output) con 80 colonne e 24 righe, con testo bianco e sfondo nero, dove verrà scritto l'output prodotto dal programma (codifica ASCII);
- una tastiera (input) con un buffer di 256 caratteri (codifica ASCII) che verrà usato per fornire l'input al programma;
- una ROM (input) su cui verrà caricato il programma (che deve essere composto da un intero 32 bit senza segno big-endian che indica la lunghezza in byte del programma succeduto dal programma stesso);
- un pulsante d'accensione (input) da premere per attivare il circuito dopo aver caricato il programma nella ROM;
- un LED (output) per determinare se il pulsante d'accensione è già stato premuto;
- un LED (output) per determinare se il programma è stato caricato in memoria.

I programmi per stampare caratteri devono scriverli, uno alla volta, nell'indirizzo 0x00100000.
Invece per prendere input da tastiera il programma deve leggere a partire dall'indirizzo 0x00100004 e assumere che ci siano caratteri disponibili finché non incontra un carattere '\0'.
La memoria utilizzabile dal programma inizia all'indirizzo 0x00200000 e finisce all'indirizzo 0x00A000000, per un totale di 8 MiB di RAM disponibile.

Per usare il circuito bisogna quindi:

- scrivere un programma in C/C++;
- compilarlo con una toolchain GCC per or1k;
- convertirlo nel formato di Logisim (ci sarà un programma per fare ciò nella cartella “util”);
- caricare l'immagine generata nella ROM;

- premere il tasto “P”;
- inserire eventuale input nella tastiera su Logisim.