Una Macchina Nozionale per Architetture dei Calcolatori come possibile collegamento tra gli insegnamenti del primo anno della laurea in Informatica

Giorgio Delzanno, Daniele D'Agostino, Giovanna Guerrini e Daniele Traversaro

DIBRIS, Università degli Studi di Genova, Italy
{name.surname}@unige.it

Sommario

In questo articolo presentiamo un esperimento basato sull'adozione del modello RAM di Cook e Reckhow come macchina nozionale per introdurre il linguaggio macchina RISC-V e di un linguaggio visuale a blocchi per avvicinare i novizi ai concetti di base dell'assembler e delle architetture di calcolo. L'esperimento si è tenuto nella prima parte del corso di Architetture dei Calcolatori nell'anno accademico 2022/23 con 320 studenti con conoscenze molto eterogenee. La scelta di combinare RISC-V, modello RAM e linguaggio a blocchi è stata dettata dal tentativo di adottare una macchina nozionale in grado di rappresentare i concetti di base di architetture load-store anche per novizi e dalla quale poter ricostruire i costrutti che nel corso di Introduzione alla Programmazione venivano nel frattempo presentati attraverso il C++.

1 Contesto e Motivazioni

Il primo semestre del primo anno del Corso di Laurea in Informatica dell'Università di Genova è organizzato intorno a tre insegnamenti fondamentali: Introduzione alla Programmazione (IP), Architetture dei Calcolatori (AC), Algebra e Logica per Informatica (ALI). I tre insegnamenti condividono un nucleo comune legato ai principi della programmazione (basso ed alto livello) e della struttura degli elaboratori. Il primo anno è da alcuni anni oggetto di studio ed applicazione di metodi didattici innovativi nel tentativo di ridurre gli abbandoni e comunque stimolare beginner senza farli scoraggiare alle prime difficoltà della materia. Tra i vari tentativi, sono state definite alcune attività ponte (per collegare ad esempio IP ad ALI). In questo ambito principi e strumenti del Pensiero Computazionale (PC) si sono rivelati molto utili in moduli integrativi per studenti alle prime armi con la programmazione. Questo articolo descrive brevemente un esperimento che cade nell'intersezione $AC \cap IP \cap ALI \cap PC$, partendo dal punto di vista di AC, vedi Fig. 1.

A partire dall'edizione 2022/23 il programma di AC segue l'approccio proposto da Patterson e Hennessy in [9] adottando RISC-V [15] come architettura di riferimento, un open standard ISA (Instruction Set Architecture), basato sul principio RISC (Reduced Instruction Set Computer) [11, 12]. In [8] Hennessy e Patterson hanno indicato RISC-V come una delle maggiori opportunità della nuova golden age delle architettture insieme a multicore e acceleratori domain specific come GPU e TPU. Questo è testimoniato anche dal fatto che MIPS Technologies, l'azienda celebre per aver ideato e sviluppato per quasi quattro decenni la famiglia di architetture MIPS, di cui detiene la proprietà intellettuale, ha annunciato l'intenzione di utilizzare RISC-V anziché MIPS, come ISA di alcune nuove CPU [10].

Dal punto di vista tecnico, RISC-V è una architettura load–store con istruzioni per accedere alla memoria (load e store tra memoria e registri) e istruzioni aritmetico-logiche che operano solo su registri. Lo schema logico utilizzato da RISC-V poggia su un array composto da registri

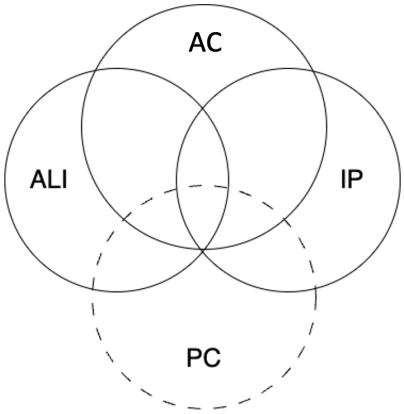

Figura 1: Relazione tra i corsi del primo anno.

per numeri interi, float e registri di controllo e stato per le modalità privilegiate che contengono informazioni sullo stato operativo del processore, i dati che vengono utilizzati in ogni istante e i meccanismi legati alla loro gestione. L'array è abbastanza ampio da ridurre al minimo la necessità di accedere alla memoria esterna per molte attività di base svolte dal processore: in questo modo si può ridurre il consumo energetico aumentando contemporaneamente le prestazioni. I progetti per i chip RISC-V sono stati per il momento ideati nelle versioni a 32, 64 e 128 bit. Il set di istruzioni è molto semplice ma allo stesso tempo estremamente modulare.

2 Assembler e Macchina Nozionale

Negli insegnamenti dedicati alle architetture dei calcolatori uno degli argomenti ritenuto ostico, in particolare da studenti con poco background informatico, è proprio quello della programmazione in linguaggio macchina. La cosa diventa ancora più evidente per modelli RISC. Infatti ridurre al minimo l'instruction set porta vantaggi in termine di efficienza e consumi ma rende molto complesso codificare anche algoritmi banali (pensate alla complessità dei programmi per le Macchine di Turing, una macchina ideale con una sola istruzione). Affrontare i concetti computazionali dovendo gestire in maniera esplicita indirizzamento e trasferimento di dati tra memoria e registri ed avendo a disposizione solo istruzioni di base (quali semplici istruzioni sia di salto che per le operazioni aritmetico-logiche che necessitano di operand già caricati nei registri) può risultare molto indigesto al primo approccio al mondo dell'informatica. Come nel caso dei linguaggi di programmazione [13, 1, 5], un modo per superare queste difficoltà iniziali può essere quello di accompagnare il discente nella costruzione e nell'affinamento del modello mentale dell'elaboratore (macchina nozionale) attraverso esperienze di apprendimento mirate a far emergere eventuali misconcezioni, cioè specifiche convinzioni scorrette responsabili di errori che mostrano l'inadeguatezza del modello mentale [13]. Una macchina nozionale, vedi Fig 2, in inglese *notional machine* (NM), è un computer idealizzato "le cui proprietà sono implicite nei costrutti del linguaggio di programmazione utilizzato" [4], ma che può anche essere reso esplicito nell'insegnamento [6]. In particolare, la NM al nostro livello di astrazione può essere vista come un'insieme di regole, non necessariamente formalizzate, per descrivere una versione astratta della vera architettura che possa aiutare uno studente a ad apprendere un modello

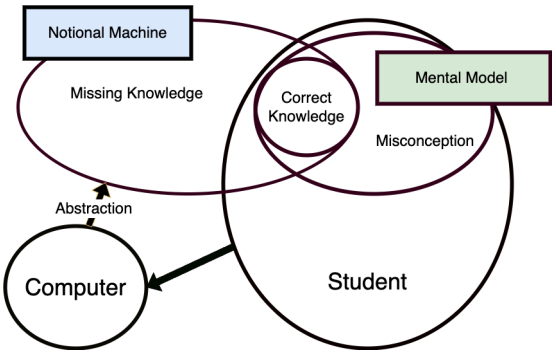

Figura 2: Macchina Nozionale e modello mentale dei discenti.

mentale più vicino possibile al suo funzionamento, minizzando ad esempio misconcezioni ed errori comuni.

3 Modello RAM e Macchina Nozionale a Blocchi

In questo breve articolo descriveremo un esperimento tenuto nella prima parte del corso AC 2022/23. Nelle prime settimane la classe era composta da più di 320 studenti con conoscenze molto eterogenee. Il rischio di scoraggiare (in maniera più o meno consapevole) molti degli indecisi presenti all'inizio del semestre è sembrato subito molto alto in particolare pensando di iniziare a parlare di assembly o di progettazione di architetture RISC. La scelta, forse inusuale, è nata sia per necessità sia dai seguenti requisiti. (1) Si è voluto adottare una NM in grado di rappresentare i concetti di base di architetture load-store e relativi reduced instruction set, partendo dalla quale poter ricostruire i costrutti che nel corso di Introduzione alla Programmazione venivano nel frattempo presentati attraverso il C++. In alternativa a modelli basati su regole o diagrammi, come quelli proposti per Python in [3], si è pensato ad un approccio, ispirato a Pensiero Computazionale e coding, in cui (2) la NM fosse esplicita ed eseguibile tramite un simulatore, (3) il simulatore stesso fosse facilmente manipolabile anche dagli studenti allo scopo di permettere di adattarlo al proprio modello mentale, confrontarlo con quello proposto dai docenti, e fare sperimentazioni su di esso.

La scelta è caduta in maniera molto naturale sulla combinazione di due mondi solo apparentemente lontani tra loro:

(M1) un'idealizzazione di un computer RISC basato sul modello delle Random Access Machine introdotte da Cook e Reckhow in [2];

(M2) la descrizione di tale macchina non tramite linguaggi formali (ad esempio transition system o sistemi induttivi) bensi tramite un linguaggio di programmazione a blocchi come Scratch[1] con semantica intuitiva e descrizione delle istruzioni anche in italiano (o altre lingue).

[1] https://scratch.mit.edu/

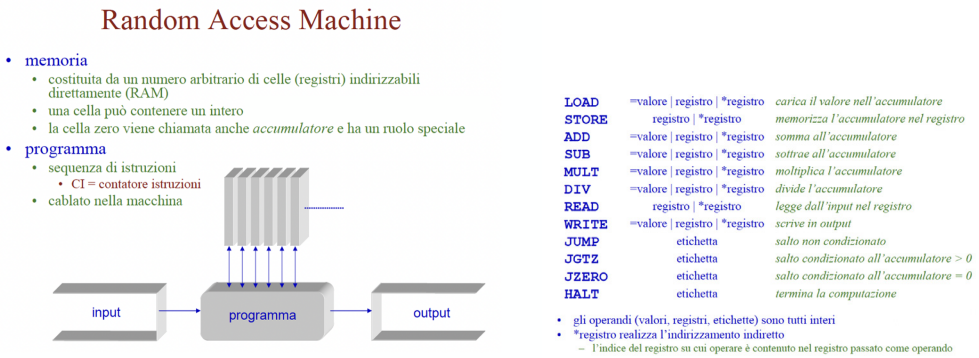

Figura 3: Modello e istruzioni RAM

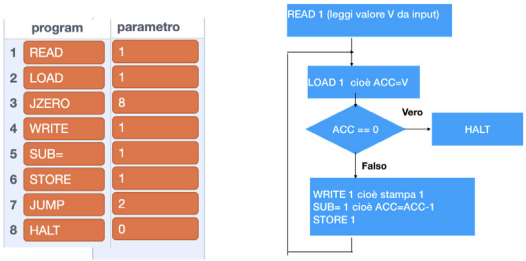

Figura 4: Esempio di programma

4 Lezioni e Compiti a Casa

Dopo una prima settimana di lezione con concetti generali sulle componenti di un elaboratore, abbiamo quindi introdotto in una lezione frontale modello e linguaggio della RAM, vedi Fig. 3, con spiegazione informale ed esempi di programmi utili in generale per capire ciclo di fetch, gestione di registri, e operazioni.

Una seconda lezione è stata quindi dedicata alla definizione in classe della macchina nozionale attraverso il linguaggio a blocchi di Scratch. In questa fase abbiamo decomposto la macchina nozionale nelle seguenti componenti: *programma*, *stato*, *ciclo di fetch*, *decodifica*, ed *esecuzione*. Un programma è stato rappresentato con due liste Scratch con istruzioni e parametri come in Fig. 4 dove abbiamo mostriamo un diagramma di flusso usato come ulteriore aiuto per comprende il collegamento con costrutti ad più alto livello. Per rappresentare i registri di stato (es. program counter e accumulatore) e i registri di memoria abbiamo usato rispettivamente variabili e liste Scratch. L'editor ha l'opzione per visualizzare lo stato di tali oggetti che naturalmente si adatta a rappresentare graficamente lo stato corrente di un programma, come in Fig. 5 in alto a sinistra, e il programma tramite liste di stringhe e parametri, vedi Fig. 5 in basso. La nozione di esecuzione è stata descritta tramite regole per definire il significato del ciclo di fetch, vedi Fig. 6 a sinistra, e regole di decodifica per le singole istruzioni, Fig. 6 a destra. Per queste regole abbiamo sfruttato i costrutti di Scratch, molto vicini a regole in linguaggio naturale ma tuttavia eseguibili come programmi imperativi. Per rappresentare l'esecuzione ci siamo basati quindi sull'interprete sottostante con la possibilità di eseguire passo per passo un programma, funzionalità programmata a sua volta con messaggi ed eventi in Scratch.

Figura 5: Stato

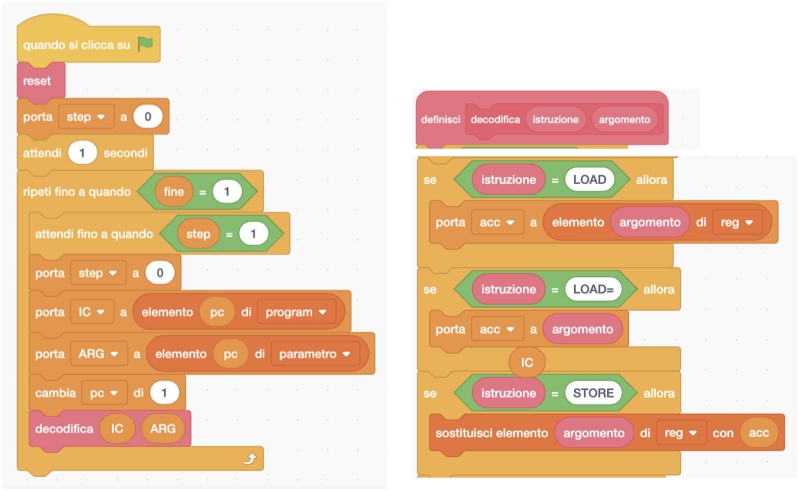

Figura 6: Rappresentazione del ciclo di fetch e regole di decodifica

Il programma risultante è stato costruito insieme agli studenti in classe illustrando principalmente programma, stato, ciclo di fetch e alcuni esempi di decodifica. In questo modo si è la possibilità ai singoli studenti di elaborare il proprio modello mentale dell'instruction set completo ed eventualmente completarlo creando un cosidetto remix del progetto Scratch. Una bozza di interprete in Scratch è stata comunque condivisa sul repository pubblico di Scratch[2] proponendo agli studenti di risolvere come esercizi asincroni, di cui due con consegna, una serie di esercizi di programmazione di difficoltà crescente alcuni con la richiesta di aggiungere nuove istruzioni (ad esempio indirizzamento indiretto) e quindi le corrispondenti regole alla notional machine per ragionare anche sul proprio modello mentale, vedi 7.

[2]https://bit.ly/ramsim

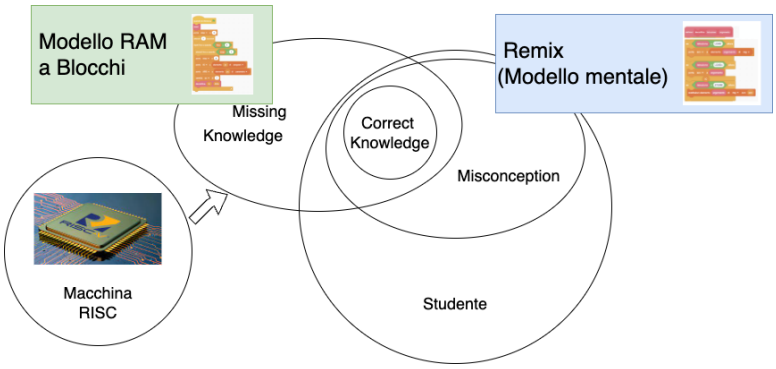

Figura 7: Macchina nozionale e modello RAM

5 Implicazioni e Discussione

Il vantaggio di questo approccio, dettato dall'esigenza di formare un terreno comune sul quale poi costruire la parte rimanente del corso, è stato molteplice. Gli studenti inesperti hanno potuto cimentarsi subito (quasi inconsciamente) con un interprete e comunque con una artefatto da usare per modellare il proprio modello mentale di calcolatore RISC. Questo a permesso di explorare misconception legate ad esempio al funzionamento di alcuni registri (accumulatore e program counter) e al funzionamento delle operazioni (ad es. eventuale reset di celle sorgenti di una load, funzionamento indirizzamento indiretto anche con celle di memoria, ecc). Il legame tra RAM e RISC-V è stato poi ulteriormente evidenziato nel corso del semestre proponendo gli stessi esercizi proposti con la macchina RAM anche in RISC-V. In questo caso è stato evidenziato come la logica risolutiva sia simile ma l'ìmplementazione per certi versi semplificata dalla possibilità di utilizzare un instruction set più articolato, per altri complicata dalla anche se ovviamente con complicazioni legate alla necessità di gestire formato, dimensione istruzioni, indirizzamento registri e memoria secondo lo standard RISC-V.

L'esperimento è risultato utile per studenti alle prime armi con programmazione e architetture anche perchè ha permesso di stabilire un ponte tra i diversi insegnamenti del primo anno incluso il corso di Algebra e Logica grazie ad esempio al concetto di macchina astratta. In linea di principio si potrebbe ulteriormente sviluppare questa idea sia provando a dedicare una lezione di ALI per la definizione formale (attraverso insiemi e relazioni) della semantica operativa di una versione core della RAM e usare formule logiche e definizione induttive induzione per specificare sue proprietà. Un approccio di questo tipo potrebbe essere di interesse anche per studenti esperti di programmazione che potrebbero cimentarsi invece sull'applicazione di concetti di algebra e logica in un dominio a loro consueto (programmazione).

Esistono esempi di linguaggi a blocchi customizzabili, ad esempio tramite la libreria Google Blockly[3], e strumenti low code che potrebbero essere utilizzati per evitare di confondere gli studenti con strumenti come Scratch nati con altri obiettivi formativi e per livelli di istruzione più bassi.

Gli studenti già esperti hanno comunque trovato stimolante questo approccio trasformando l'esperimento proposto in un progettino per la creazione di un interprete per la RAM con linguaggio di sviluppo scelto a piacere (es. Python, Rust) trovando stimoli ulteriori rispetto agli esercizi tradizionali proposti nelle prime settimane di IP.

[3] https://developers.google.com/blockly

Visto il ridotto set di istruzioni fornito dalla RAM, per tutti gli studenti, esperti e non, la programmazione di soluzioni di problemi apparentemente semplici si è rivelato un problema concettualmente interessante. Un esempio per tutti è il problema di effettuare operazioni aggregate su K numeri letti da input dove K è a sua volta un input. Questo problema, riproposto poi anche in assempler RISC-V alla fine del semestre, è uno spunto per confrontarsi con problematiche di gestione dinamica della memoria (ad esempio attraverso indirizzamento indiretto).

Nel prossimo anno accademico prevediamo di estendere tali attività per creare un ponte con insegnamenti di anni succeddivi, specificamente Programmazione Concorrente ed Algoritmi Distribuiti ed High Performance Computing. Il focus del primo anno di un corso di studio in Informatica dev'essere necessariamente quello di dotare gli studenti degli strumenti e conoscenze necessarie a scrivere programmi corretti. Questo ovviamente non è sufficiente, ma dev'essere accompagnato dall econoscenze e tecniche necessarie a scrivere programmi corretti e perfomanti.

Il modello sequenziale è stato superato dal 2006, con l'avvento dei primi processori dual core. Il corso di AC rappresenta il luogo naturale in cui insegnare come questa evoluzione condizioni la scrittura di programmi efficienti in quanto in grado di sfruttare efficacemente la cache, le unità di vettorizzazione e la presenza di più core [14]. Tale approccio richiederà sia il passaggio dal modello RAM al modello MP-RAM [7], sia l'introduzione di esercitazioni mirate alla valutazione approfondita delle prestazioni utilizzando opportuni strumenti di profiling [4].

Riferimenti bibliografici

[1] Benedict Du Boulay. Some difficulties of learning to program. *Journal of Educational Computing Research*, 2(1):57–73, 1986.

[2] Stephen A. Cook and Robert A. Reckhow. Time-bounded random access machines. In Patrick C. Fischer, H. Paul Zeiger, Jeffrey D. Ullman, and Arnold L. Rosenberg, editors, *Proceedings of the 4th Annual ACM Symposium on Theory of Computing, May 1-3, 1972, Denver, Colorado, USA*, pages 73–80. ACM, 1972.

[3] Paul E. Dickson, Tim D. Richards, and Brett A. Becker. Experiences implementing and utilizing a notional machine in the classroom. In Larry Merkle, Maureen Doyle, Judithe Sheard, Leen-Kiat Soh, and Brian Dorn, editors, *SIGCSE 2022: The 53rd ACM Technical Symposium on Computer Science Education, Providence, RI, USA, March 3-5, 2022, Volume 1*, pages 850–856. ACM, 2022.

[4] Benedict Du Boulay. Some difficulties of learning to program. *Journal of Educational Computing Research*, 2(1):57–73, 1986.

[5] Benedict du Boulay, Tim O'Shea, and John Monk. The black box inside the glass box: presenting computing concepts to novices. *International Journal of Man-Machine Studies*, 14(3):237–249, 1981.

[6] Benedict Du Boulay, Tim O'Shea, and John Monk. The black box inside the glass box: presenting computing concepts to novices. *International Journal of man-machine studies*, 14(3):237–249, 1981.

[7] Steven Fortune and James Wyllie. Parallelism in random access machines. In *Proceedings of the tenth annual ACM symposium on Theory of computing*, pages 114–118, 1978.

[8] John L. Hennessy and David A. Patterson. A new golden age for computer architecture. *Commun. ACM*, 62(2):48–60, 2019.

[9] John L. Hennessy and David A. Patterson. *Struttura e progetto dei calcolatori. Progettare con RISC-V*. Zanichelli, 2023.

[10] Steven Leibson. Mips joins the risc-v gang. *Forbes*, 2022.

[4] https://book.easyperf.net/perf_book

[11] David A. Patterson. Reduced instruction set computers then and now. *Computer*, 50(12):10–12, 2017.

[12] David A. Patterson. 50 years of computer architecture: From the mainframe CPU to the domain-specific tpu and the open RISC-V instruction set. In *2018 IEEE International Solid-State Circuits Conference, ISSCC 2018, San Francisco, CA, USA, February 11-15, 2018*, pages 27–31. IEEE, 2018.

[13] Juha Sorva. Notional machines and introductory programming education. *ACM Trans. Comput. Educ.*, 13(2), jul 2013.

[14] Herb Sutter et al. The free lunch is over: A fundamental turn toward concurrency in software. *Dr. Dobb's journal*, 30(3):202–210, 2005.

[15] Andrew Waterman, Yunsup Lee, Rimas Avizienis, Henry Cook, David A. Patterson, and Krste Asanovic. The RISC-V instruction set. In *2013 IEEE Hot Chips 25 Symposium (HCS), Stanford University, CA, USA, August 25-27, 2013*, page 1. IEEE, 2013.