Università di Bergamo

Relazione progetto

Linguaggi Formali e Compilatori

Parser ISA Risc-V

Autore:
Dario Sardi

Supervisore:
Giuseppe Psaila

12 Marzo 2019

Abstract

Il progetto è volto allo sviluppo di un traduttore di un subset di istruzioni appartenenti all'ISA RISC-V da linguaggio assembly a linguaggio naturale.

1 Output e Input

Il programma prende come input il testo contenuto nel file "resources/input.file" e restituisce la traduzione in linguaggio naturale come output su terminale seguito da una eventuale lista di errori.
Un esempio di output privo di errori:

```
----- PARSING STARTED-------
created ADDI between 1 and 12 to be saved at 10
created OR between 12 and 2 to be saved at 5
added label li
created ADD between 7 and 5 to be saved at 22
created AND between 7 and 5 to be saved at 23
label check for li
jump to line 3 if condition is true
created OR between 22 and 23 to be saved at 24
----- PARSING DONE -------
```

In caso di errori l'output è il seguente:

```
----- PARSING STARTED-------
created ADDI between 1 and 12 to be saved at 10
created OR between 12 and 2 to be saved at 5
added label li
created ADD between 7 and 5 to be saved at 22
created AND between 7 and 45 to be saved at 23

##############################
#  ERROR! PARSING STOPPED!  #
##############################

ERRORS:
1. label 'li' at line 4 already exist, ignoring it.
2. Lexer error at [6:15].Max register value is 30!
```

2 Registri e immediati

Per le specifiche scelte per questo progetto si prevede l'esistenza di 30 registri indirizzabili attraverso la sequenza '0xNumeroDiRegistro' dove il numero del registro è un numero intero compreso fra 0 e 30.
La scelta del prefisso è stata dettata dalla necessità di distinguere numeri interi e numeri dei registri durante il parsing e rendere il meno confusionario possibile il codice.
Per quanto riguarda gli immediati è stata prevista la dimensione massima di una word su ISA RISC-V equivalente al valore decimale 4096.

3 Variabili

È previsto l'utilizzo di variabili statiche per etichettare registri o valori notevoli. La sintassi per dichiarare il nome di un registro è:

```
DR etichetta 0xNumeroDiRegistro
```

Una definizione valida puo dunque essere

```
DR RegistroA 0x9
```

Si ricorda il valore massimo assegnabile a un registro pari a 0x30 per specifiche precedentemente definite. Per quanto riguarda le variabili, si possono creare variabili numeriche statiche di tipo Byte e Word. Risulta possibile riservare una variabile senza inizializzarla attraverso il comando RESB per riservare un Byte e RESW per riservare una Word. Esempio di variabili riservate sono

```
RESW valoreMassimo
RESB threshold
```

Questa operazione è stata introdotta per una implementazione futura dove variabili dinamiche potranno essere assegnate o variate utilizzandole come destinazione all'interno di funzioni. Il comando attualmente utile risulta essere la dichiarazione di una variabile con relativa inizializzazione effettuabile tramite

```
DW valoreMassimo 3623
DB threshold 90
```

3.1 Error handling

Nel caso di una dichiarazione di variabile troppo grande il programma risponderà settando la variabile a 0 e proseguendo normalmente.
Esempi di input-output possibili sono

1: input

```
1    DW ciao 5400
```

2: output

```
1    ----- INIZIO PARSING -------
2    defined new Word-type variable with value 0
3    named ciao
4    ----- FINE PARSING -------
5    1.  variable value is not Word type
6        (max value 4096), ciao set to 0
```

Oppure

3: input

```
1    DB ciao 300
```

4: output

```
1    ----- INIZIO PARSING -------
2    defined new Byte-type variable with value 0
3    named ciao
4    ----- FINE PARSING -------
5    1.  variable value is not Byte type
6        (max value 256), ciao set to 0
```

4 Le funzioni

Il subset di istruzioni considerato prevede due tipologie di funzioni, le R-type e le I-type.
In entrambe le funzioni la struttura sintattica risulta essere

```
    Funzione Destinazione Operando1 Operando2
```

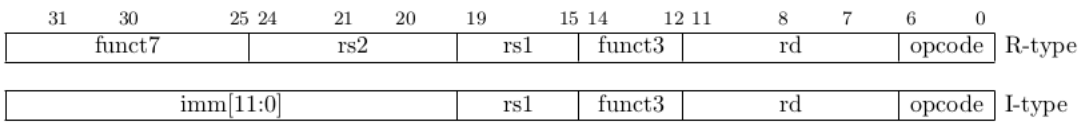

Figure 1: Struttura in binario delle funzioni e relativi attributi

Per le funzioni R-type destinazione e operandi saranno registri

```
Funzione RegDestinazione RegOperando1 RegOperando2
```

Per le funzioni di I-type il secondo operando dovrà essere un immediato

```
Funzione RegDestinazione RegOperando1 ImmOperando2
```

Le funzioni previste sono le seguenti:

1. R-Type: ADD,SUB,MUL,OR,XOR,AND
2. I-Type: ADDI,SUBI,ORI,XORI,ANDI

Esempi di funzioni sono

5: input

```
1  ADDI 0x10 0x01 12
2  OR 0x05 0x12 0x02
```

6: output

```
1 ----- INIZIO PARSING -------
2 created ADDI between 1 and 12 to be saved at 10
3 created OR between 12 and 2 to be saved at 5
4 ----- FINE PARSING -------
```

4.1 Error Handling

5 Loop

5.1 Error Handling