UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

> Reinventa el mundo <

COMPUTER SYSTEMS

RISC V ASSEMBLY

TRANSFORMATEC

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

> Reinventa el mundo <

Goals

Analyze the assembly language and the instruction set architecture of RISC V

TRANSFORMATEC

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

> Reinventa el mundo <

Summary

1. Conditional Statements
2. Loops
3. Arrays
4. Conclusions

TRANSFORMATEC

Summary

Motivation: Programs at high-level languages can be executed in modern processor systems.

Problem: We need to understand the machine code and define its relationship with programming languages constructs.

Overview:

- Overview of conditional statements, loops, arrays and function calls.
- ARM Assembly programming for high-level constructs.
- Code and execute with an ARM emulator.

Conclusion: We can create complex programs using assembly language by defining correct machine code instructions.

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

1.

Conditional Statements

TRANSFORMATEC

If Statement

High level Code

```
if(a == m)
   f = g+h;
a = m - h;
```

RISC V Assembly Code

If Statement

High level Code

```
if(a == m)
   f = g + h;
a = m - h;
```

RISC V Assembly Code

```
#s0 = a, s1 = m
#s2 = f, s3 = g, s4 = h

BNE s0, s1, L1
ADD s2, s3, s4
L1:
SUB s0, s1, s4
```

Assembly tests opposite case (i != j) of high-level code (i == j)

If/Else Statement

High level Code

```
if(a == m)
   f = g+h;
else
   a = m - h;
```

RISC V Assembly Code

If/Else Statement

High level Code

```
if(a == m)
   f = g + h;
else
   a = m - h;
```

RISC V Assembly Code

```
#s0 = a, s1 = m
#s2 = f, s3 = g, s4 = h

   BNE s0, s1, L1
   ADD s2, s3, s4
   j L2
L1:
   SUB s0, s1, s4
L2:
```

Switch/Case Statement

High level Code

```
switch(button){
   case 1: am = 20; break;
   case 2: am = 50; break;
   case 3: am = 100; break;
   default: am = 0;
}
```

RISC V Assembly Code

Switch/Case Statement

High level Code

Equivalent with if/else

```
if          (button == 1)   am = 20;
else if     (button == 2)   am = 50;
else if     (button == 3)   am = 100;
else                        am = 0;
```

RISC V Assembly Code

Switch/Case Statement

High level Code

```
switch(sim){
   case 1: am = 20; break;
   case 2: am = 40; break;
   case 3: am = 60; break;
   default: am = 0;
}
```

RISC V Assembly Code

```
#s0 = sim, s1 = am

case1:
   ADDI t0,zero, 1
   BNE s0, t0, case2
   ADDI s1,zero,20
   j done

case2:
   ADDI t0,zero, 2
   BNE s0, t0, case3
   ADDI s1,zero,40
   j done

case3:
   ADDI t0,zero, 3
   BNE s0, t0, def
   ADDI s1,zero,60
   j done

def:
   ADD s1,zero,zero
done:
```

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

2.

Loops

TRANSFORMATEC

While Loop

High level Code

```
int pow = 1
int x = 0;

while(pow != 128){
   pow = pow*2;
   x = x + 1;
   }
```

What does the code do?

RISC V Assembly Code

While Loop

High level Code

```
int pow = 1
int x = 0;

while(pow != 128){
   pow = pow*2;
   x = x + 1;
   }
```

What does the code do?

RISC V Assembly Code

```
#s0 = pow, s1 = x

   ADDI s0,zero,1
   ADDI s1,zero,zero
   ADDI t0,zero,128
while:
   BEQ s0,t0,done
   SLLI s0,s0,1
   ADDI s1,s1,1
   J while
done:
```

Assembly tests for the opposite case (pow == 128) of the C code (pow != 128).

For Loop

For (initialization; condition; loop operation) statement

- **Initialization:** executes before the loop begins.
- **Condition:** is tested at the beginning of each iteration.
- **Loop operation:** executes at the end of each iteration.
- **Statement:** executes each time the condition is met.

Do/While Loop

High level Code

```
int pow = 1
int x = 0;

do{
   pow = pow*2;
   x = x + 1;
   } while(pow != 128);
```

RISC V Assembly Code

Do/While Loop

High level Code

```
int pow = 1
int x = 0;

do{
   pow = pow*2;
   x = x + 1;
   } while(pow != 128);
```

RISC V Assembly Code

```
#s0 = pow, s1 = x

   ADDI s0,zero,1
   ADD s1,zero,zero
   ADDI t0,zero,128

while:
   SLLI s0,s0,1
   ADDI s1,s1,1
   BNE s0,t0,while
done:
```

For Loop

High level Code

```
int sum = 0;
int i;

for(i = 0; i < 10; i = i + 1){
   sum = sum + i;
   }
```

RISC V Assembly Code

For Loop

High level Code

```
int sum = 0;
int i;

for(i = 0; i < 10; i = i + 1){
   sum = sum + i;
   }
```

RISC V Assembly Code

```
#s0 = i, s1 = sum

   ADDI s1,zero,0
   ADDI s0,zero,0
   ADDI t0,zero,10
for:
   BGE s0,t0,done
   ADD s1,s1,s0
   ADDI s0,t0,1
   J for
done:
```

For Loop

High level Code

```
int sum = 0;
int i;

for(i = 0; i < 10; i = i + 1){
   sum = sum + i;
   }
```

RISC V Assembly Code

```
#s0 = i, s1 = sum

   ADDI s1,zero,0
   ADDI s0,zero,0
   ADDI t0,zero,10
for:
   BGE s0,t0,done
   ADD s1,s1,s0
   ADDI s0,t0,1
   J for
done:
```

How to do a decremental loop?

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

3.

Arrays

TRANSFORMATEC

Arrays

Access large amounts of similar data:

- Index: access to each element
- Size: number of elements

Example: 5-element array

Base address = 0x14000000
(address of first element, scores[0])

Array elements accessed relative to base address.

For Loop to access an array

High level Code

```
int i;
int scores[200];

for(i = 0; i < 200; i = i + 1)

   scores[i] = scores[i] + 10
```

RISC V Assembly Code

```
#s0 = scores base address, s1 = i

ADDI s1,zero,0
ADDI t2,zero,200

for:
BGE s1,t2,done
SLLI t0,s1,2
ADD t0,t0,s0
LW t1, 0(t0)
ADDI t1,t1,10
SW t1, 0(t0)
ADDI s1,s1,1
J fo
done:
```

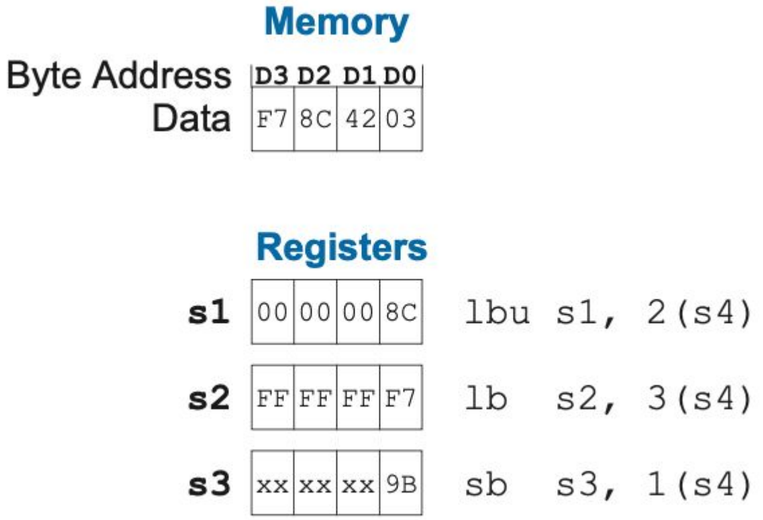
ASCII Code and Cast of Characters

- American Standard Code for Information Interchange (ASCII)
- Each text character has unique byte value
  - For example, S = 0x53, a = 0x61, A = 0x41
  - Lower-case and upper-case differ by 0x20 (32)

#	Char	#	Char	#	Char	#	Char	#	Char	#	Char
20	space	30	0	40	@	50	P	60	`	70	p
21	!	31	1	41	A	51	Q	61	a	71	q
22	"	32	2	42	B	52	R	62	b	72	r
23	#	33	3	43	C	53	S	63	c	73	s
24	$	34	4	44	D	54	T	64	d	74	t
25	%	35	5	45	E	55	U	65	e	75	u
26	&	36	6	46	F	56	V	66	f	76	v
27	'	37	7	47	G	57	W	67	g	77	w
28	(	38	8	48	H	58	X	68	h	78	x
29	)	39	9	49	I	59	Y	69	i	79	y
2A	*	3A	:	4A	J	5A	Z	6A	j	7A	z
2B	+	3B	;	4B	K	5B	[	6B	k	7B	{
2C	,	3C	<	4C	L	5C	\	6C	l	7C	|
2D	-	3D	=	4D	M	5D	]	6D	m	7D	}
2E	.	3E	>	4E	N	5E	^	6E	n	7E	~
2F	/	3F	?	4F	O	5F	_	6F	o		

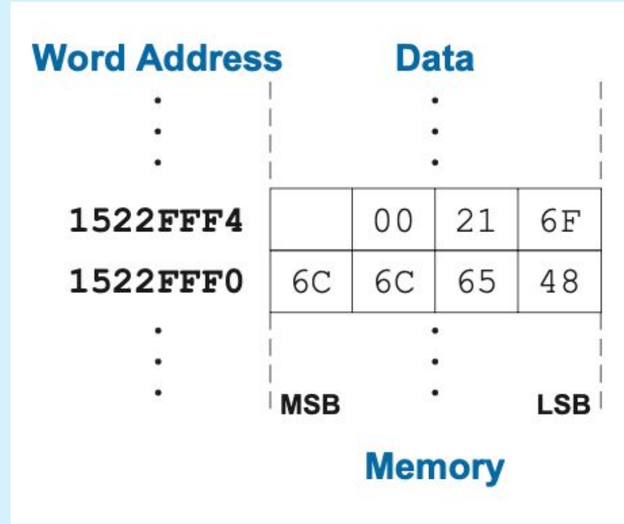
Load & Store in an array

- RISC-V also defines lh, lhu, and sh half-word loads and stores that operate on 16-bit data.
- Memory addresses for these instructions must be half-word aligned



“Hello!”

H = 0x48
e = 65
l = 6C
l = 6C
o = 6F
! = 21
Null = 00



UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Conclusions

- We reviewed fundamentals concepts in programming languages constructs.
- We reviewed assembly implementation for conditional statements, loops, arrays.
- We coded and executed high-level constructs using ARM syntax and instructions.
- We conclude that complex programs have a direct implementation in machine code that allows to execute them in the processor.

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Reference Books

- Patterson, D. A., & Hennessy, J. L. (2020). Computer Organization and Design RISC-V Edition: The Hardware Software Interface. Morgan Kaufmann
- “The elements of computing systems: building a modern computer from first principles” Nisan, N., & Schocken, S. (2021). MIT press
- Silberschatz, A., Gagne, G., & Galvin, P. B. (2015). Operating system concepts (9th edition, international student version). John Wiley & Sons Inc.
- ”Digital Design and Computer Architecture, RISC-V Edition”. Morgan Kaufmann. Harris, S., & Harris, D. (2021).

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

> Reinventa el mundo <

Questions?

TRANSFORMATEC