NTNU

Innovation and Creativity

Introduction to the Dark Assembly

Alexandru Iordan

Institutt for datateknikk og informasjonsvitenskap

Plan

- What is Dark?
- Assembly vs. Java
- Dark stack machine
  - Overview of the stack machine
  - Dark stack machine instance
- Dark Load-Store machine
  - Overview of the Load-Store machine
  - Arithmetic
  - Memory
  - Control Flow
- Some tips

NTNU
Innovation and Creativity

What is Dark?

- Dark is a computer architecture simulator
- You can write programs in assembly language and test their execution on accumulator machine, stack machine, load-store machine, memory-memory machine or virtual machine
- In AoC you will only be tested in two of the architectures: Load-Store and Stack architecture

NTNU
Innovation and Creativity

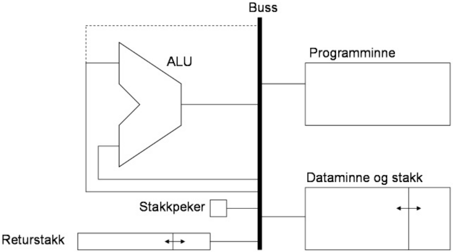
Assembly vs. Java

- Assembly instructions are low-level
- Writing assembly requires knowledge of the hardware machine
- Assembly runs only on a given machine, while Java can run on any machine
- A line in Java can correspond to several dozen lines of assembly

NTNU
Innovation and Creativity

Overview of the stack machine

- All instructions manipulate a stack
- All operations take place at the top of the stack
- Arithmetic is done by popping the top two items of the stack and pushing back the result



Arithmetic

Common arithmetic / logic operations:

- **add**
- **sub**
- **div**
- **mul**

Example:

```
push 3  ; 3 is placed on the stack
push 5  ; 5 is placed on the stack
add     ; The two above are popped and the result is
          pushed on the stack
```

Stack manipulation

- **dup** - copy the top item and puts it on top
- **swap** - swaps the order of the top two elements
- **drop** - the top item is taken away
- **rot** - the top three values are rotated so that the value that was on the top, is now located at the bottom

NTNU
Innovation and Creativity

How to perform a conditional jump?

It requires 2 steps:

1. a test is performed on the two top elements of the stack and the stack is updated with the result of the test(1 for true and 0 for false)
2. a jump is performed considering this result

Available tests:

- **eq** - EQual
- **ge** - Greater or Equal
- **gt** - Greater Than
- **le** - Less or Equal
- **lt** - Less Than
- **ne** - Not Equal

Order: $result \leftarrow stack[top - 1] \oplus stack[top]$

NTNU
Innovation and Creativity

How to perform a jump? - II

The jump is based on the test:

- **jfalse** - Jump if 0 is on the stack
- **jtrue** - Jump if 1 is on the stack
- **jmp** - Jump no matter

The pairs eq / jtrue and ne / jfalse are equivalent!

Example:

```
start
  lt          ; compares the top two elements of the stack
  jtrue end   ; if stack[top-1]< stack[top] jump to end
  jmp start   ; jump back to start
end
```

Dark stack machine example

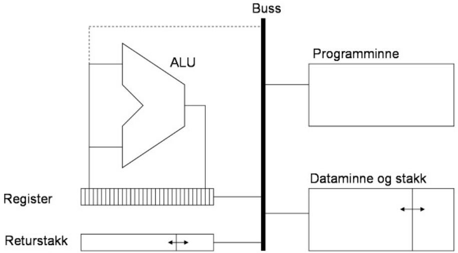
— We will solve the "Min" assigement (272) from AoC

"The task is to find the smallest value in a list of numbers using a Dark stack machine.
When the task is verified, the stack contains a list of numbers. At the bottom of the stack is the number 0 indicating the end of the list.
Once the program has finished running, only one value should be left on the stack: the minimum value from the original list."

Overview of the Load-Store machine

In a load-store machine the following resources are available:

- 32 registers ($0 to $31), however:
  - register 0 ($0) is always 0
  - register 1 ($1) is reserved for assembler
  - register 31 ($31) is used for the stack
- one large memory



Arithmetic I - Add

Syntax

add REGISTER_A,REGISTER_B,{REGISTER_C|NUMBER}

Semantics register_a $\leftarrow$ register_b + {register_c |NUMBER}

Register_a receives the sum of the second and final values.

Example:

```
load $2, 10    ; load the value 10 into $2
add $3, $2, 10 ; sum of $2 and 10 is stored in $3
add $4, $2, $3 ; sum of $2 and $3 is stored in $4
```

What value is now stored in register 4?

NTNU
Innovation and Creativity

Arithmetic II - Sub

Syntax:

sub REGISTER_A,REGISTER_B,{REGISTER_C|NUMBER}

Semantics:

register_a ← register_b − {register_c|NUMBER}

Register_a receive the difference of the second and final values.

Example:

```
sub $1, $2, $3 ; $1 = $2 - $3
```

NTNU
Innovation and Creativity

And so on ...

- Multiplication, Division, logical operators (AND, OR, SHIFT,etc.) are equivalent
- Read the documentation (AoC)
- Particularly interesting instructions:
  - **inc** - increment a register by 1
  - **dec** - decrement a register by 1
  - Useful in loops

NTNU
Innovation and Creativity

A small example

ASSIGNMENT: r8 = (r1 * r2) + (r3 * r4)

Java

```
int temp1 = r1 * r2;
int temp2 = r3 * r4;
r8   = temp1 + temp2;
```

DARK

```
mul    $5, $1, $2
mul    $6, $3, $4
add    $8, $5, $6
```

NTNU

Innovation and Creativity

Memory I - Load

Syntax:

`load` REGISTER_A,{REGISTER_B+NUMBER|NUMBER|VARIABLE}

Semantics:

register_a ← {mem[register_b+NUM]|NUM|VAR}

Register_a receives either the value specified as a number, the contents of variable or the contents of memory location that is found from register_b + number.

Example:

load $4, $3+0 ; The contents of memory address $3 is loaded in register $4

NTNU
Innovation and Creativity

Memory II - Store

Syntax:

`store` REGISTER_A,{REGISTER_B+NUMBER|VARIABLE}

Semantics:

{mem[register_b+NUM]|VAR} ← register_a
The register_a is stored in memory location pointed by register_b + number or by a variable.

Example:

store $2, $3+0 ; Register $2 is stored in the memory location pointed by $3.

Control Flow I - JMP

The program jumps to a given label in the code.
Example:

```
top:
          inc   $7
          jmp   top
```

Control Flow II - Conditional jump

There are many situations where you just want to jump in given situations. These are the conditional jumps:

- **jeq** - Jump if EQual
- **jne** - Jump if Not Equal
- **jge** - Jump if Greater or Equal
- **jgt** - Jump if Greater
- **jle** - Jump if Less or Equal
- **jlt** - Jump if Less

```
load $3, 5
start:
  inc $2
  jeq $2, $3, stop
  jmp start
stop:
```

NTNU
Innovation and Creativity

IF statements

In Java and similar languages, we often need to express conditions in the form of IF sentences. One example:

Java

```
if (a<b) {
  a = a + b;
} else {
  a = a - b;
}
```

DARK

```
            ; Note: a=$2, b=$3
            jge   $2, $3, else
            add   $2, $2, $3
            jmp   end
else:
            sub   $2, $2, $3
end:
```

NTNU
Innovation and Creativity

WHILE loops

Java

```
int a = 0;
while(a <= 10) {
  a = a + 1;
}
```

DARK

```
      ; Note: a=$5,$6=10
      load $5, $0
      load $6, 10
start:
      jgt $5, $6, end
      inc $5
      jmp start
end:
      ...
```

FOR loops

Java

```
int j = 5;
for(int i = 0; i <= j; i++){
  // something to do
}
```

DARK

```
        load $5, 0   ; i=0
        load $6, 5   ; j=5
start:
        jgt $5, $6, end
        ; something to do
        inc $5
        jmp start
end:
```

Note that the assembly code for a for-loop is identical to the while-loop of the previous example.

Some tips

- Remember to remove the test data before validating!
- After you solve a dark subject, you can find your code and solution suggestions in the task book.
- Use the comments frequently!
- Start with a small problem or part of the problem. This way you do not have to think about everything at once.
- Take a piece of paper and draw the stack so you find out what you must do before you write code.
- Keep it simple!

NTNU
Innovation and Creativity