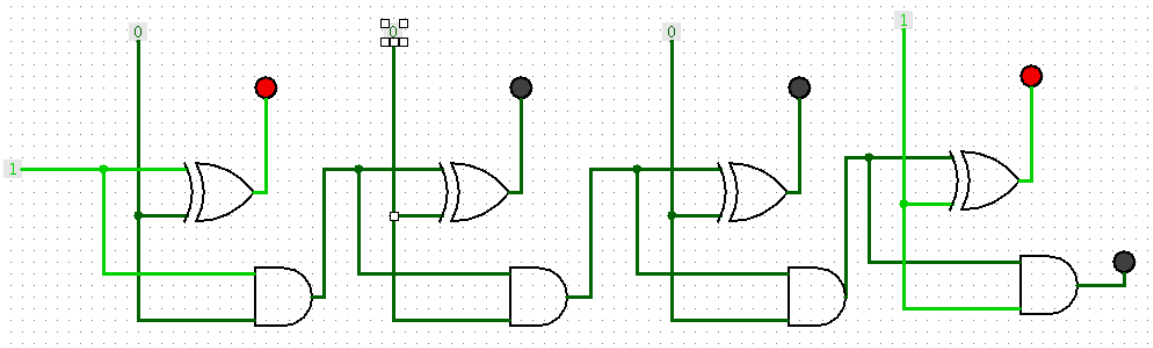
Essential Skills: Assignment Computer architecture

Ali Abdulmadzhidov

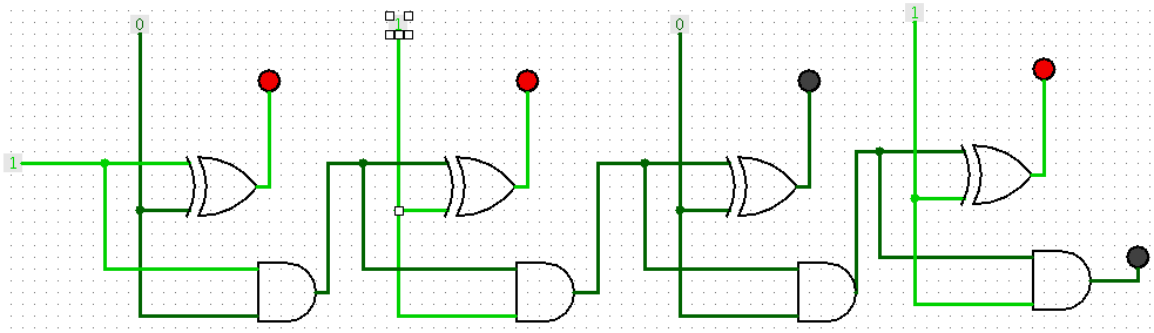
2 October 2016

1 Digital circuit

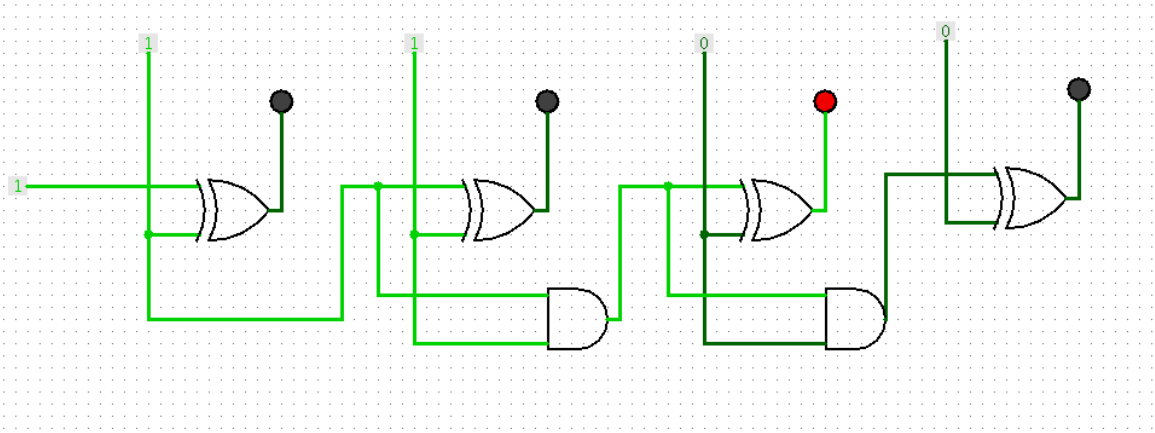
We add 1000 (right to left) from constant's. Output on leds is 1001.



We add 1010 (right to left) from constant's. Output on leds is 1011



Same with 6 elements



2 Perfomance measurement

2.1 Installation SIM-PL

1. Downloading SimPL and components

```
wget https://staff.fnwi.uva.nl/t.r.walstra/innopolis/SIM-PL_2.3.0.zip
wget https://staff.fnwi.uva.nl/e.h.steffens/Downloads/Arco/Innopolis/ComponentLab1
```

2. Unzipping

```
unzip SIM-PL_2.3.0.zip -d SIMPL
unzip components_lab_1.zip -d comps
```

3. Executing executor

```
cd SIMPL
java -jar executor.jar
```

4. File-Open needed architecture. Than in code viewer file-open needed asm program.

2.2 Measuring parameters.

2.2.1 Singlecycle. Assume that we have 10hz clock rate

2.2.2 ForLoop.asm

1. 45 instructions
2. CPI - 1
3. CpuTime = 4.5sec.

2.2.3 Addition.asm

1. 6 instructions
2. CPI - 1
3. CpuTime = 0.6sec

2.2.4 Square.asm

1. 99 instructions
2. CPI - 1
3. CpuTime = 9.9sec

2.2.5 Multicycle. Assume that we have 50hz clock rate

2.2.6 ForLoop.asm

1. 46 instructions
2. CPI - 3.8
3. CpuTime = 3 sec

2.2.7 Addition.asm

1. 6 instructions
2. CPI - 3.6
3. CpuTime = 0.44 sec

2.2.8 Square.asm

1. 117 instructions
2. CPI - 3.5
3. CpuTime = 7.08 sec

2.2.9 Pipelined Assume that we have 50hz clock rate

2.2.10 ForLoop.asm

1. 25 instructions
2. CPI - 1
3. CpuTime = 0.5 se

2.2.11 Addition.asm

1. 108 instructions
2. CPI - 1
3. CpuTime = 2.16 sec

2.2.12 Square.asm

1. 128 instructions
2. CPI - 1
3. CpuTime - 2.56 sec

2.3 Explain the following

1. addition.asm was performed good by three archs. In forloop and square pipelined was the best, cause it can it has shorter cycle and can divide instrcutions to execute them in one time.
2. program counter shows memory adress of instruction that is executing now or'll be executed next
3. multicycle take more then one cycle to execute instruction.

2.4 Explain the following

1. ADDI - adds a register and a sign-extended immediate value and saves result in another register
2. NOP - command that does nothing. Used to make delay or to force memory alignment.
3. LI - puts number to register.

```
li $s0, 123 # puts 123 to #s0
```

4. BNE - jumps to label

```
label:  li $s0, 0
        BNE label
```

2.5 Subtracting 2 variables

```
# Generated assembly code
@include "Mips.wasm"

#
.data MyRegisters : REGISTERS
0x00 : WORD 0x00 # put $0 to zero (default assumption)

.data MyMemory : DATAMEM
0x00 : WORD 0x00

# De instructiecodes worden in
.code MyCode: MIPS, MyMemory

# Sla getal 1 op in register 1
LI      $4, 1
ADD     $1, $4, $0

# Sla getal 2 op in register 2
LI      $5, 2
ADD     $2, $5, $0

# Tel register 1 en 2 bij elkaar op
# en plaats het resultaat in register 3
SUB     $6, $1, $2
ADD     $3, $6, $0

# - Orginal C code -
#{
#       int a;
#       int b;
#
#       int result;
#
#       a = 1;
#       b = 2;
#
#       result = a - b;
#
#}
```