Computer Structure Lab - Fall 1998
Dr. Guy Even

Handout #6: The simplified DLX - implementation and testing (v 1.0)

1 Functional Units

There are two non-trivial functional units in the datapath: the ALU and the shifter.

1.1 The ALU

The 32-bit adder in the ALU can be designed using ABEL. However, you may also use three 16-bit adder/subtractors from the Xilinx library (ADSU16) to build a 32-bit conditional sum adder. The usage of the adder/subtractor eliminates the need to invert the second argument in a subtraction instruction. Care should be taken in the implementation of the ALU to reduce the latency since it lies on the critical path. Moreover, a costly design might create problems in the place and route stage.

The ALU supports bitwise logical instructions and comparison instructions. A comparison is implemented by a subtraction followed by examining the sign of the result and whether the result equals zero. More implementation details appear in the book of Müller and Paul.

1.2 The Shifter

The shifter is a 32-bit left/right logical shifter. This means that a zero is pushed in from the right (left) in case of a left (right) shift. The control inputs of the shifter are: `shift` and `right`. The `shift` input indicates whether a shift should take place (otherwise the output equals the input). The `right` signal indicates whether the shift is a right shift. (Why do we want the shifter to support also the identity function?)

2 Special Instructions

The DLX assembly language includes for convenience two additional instructions which are aliases for DLX instructions.

1. MOVE. A move instruction has two arguments (e.g. `move RD RS1`), and is an alias for `addi RD RS1 0`.
2. NOP. A no-operation instruction has no arguments and is an alias for `addi R0 R0 0`.

3 The Main Memory Map

3.1 Physical address space

The RESA main memory physical address space is divided as follows:

1. 0x00000000 - 0x0000ffff : 64KW of ROM. This ROM stores a test program for the DLX design.
2. 0x00010000 - 0x007fffff : This address space is not allocated to any device.
3. 0x00800000 - 0x008fffff: 1MW of RAM. This space is used for the user programs.
4. Some of the other address space is allocated to various devices (CPU board, monitor slave, etc.). *Do not attempt to access addresses in this space since it might damage the RESA.*

3.2 Translating logical addresses to physical addresses

The logical address space of the DLX processor is simply 32-bits wide (i.e. $[0,(2^{32}-1)]$). As described above, the physical address space 0x00000000 - 0x0000ffff is used for a ROM test program, whereas the physical address space 0x00800000 - 0x008fffff allows for 1 MW for user programs.

Suppose we want to have the uniform abstraction that, regardless of whether the test program or a user program is running, the address space starts at 0x00000000. How do we supply this abstraction? A translation is required from logical addresses to physical addresses that the running program (test program or user program) is unaware of.

The translation is done by a Memory Mapping Unit (MMU) that maps a physical address to a logical one. The inputs to this mapping are the AO (address out) bus of the DLX and a `mem_map` signal. The output of the MMU is the `MMU_AO` bus that connects to the I/O control logic. The `mem_map` signal tells the MMU how to translate the logical address into a physical one.

We suggest that the `mem_map` signal be taken from bit 23 of the command register. This bit controls the functionality of the MMU as follows: If a test program is running, then `MMU_AO`=`AO`. If a user program is running, then $\texttt{MMU\_AO} = \texttt{AO} + 0x00800000$. You may assume that the user program's logical address space is limited to $[0,2^{23}-1]$ to simplify the MMU design.

4 The Test Program

The test program consists of nine test procedures. The source of the test program is given in the appendix. An outline is depicted below.

```
*-------------------------------------------------------------------*
* Example of a test program for the DLX
*-------------------------------------------------------------------*
...

        pc = 0x00000000
test0:  ...
```

```
RESA:   dc 0x52455341          * "RESA"
USER:   dc 0x00800001          * Start address of user program
RAM:    dc 0x00800000

test1:  ...
test8:  dc 0xc0000008          * Special-NOP

        lw R30 R0  RAM         * R30 := Address of RAM
        lw R1  R0  RESA
        lw R2  R30 0x0         * read from M[0x800000]
        xor R2 R2 R1           * if (M[0x800000]=="RESA")
        beqz R2 start          *   then jump to user program
        bnez R2 ramtst         *   else perform memory test
        ...

start:  lw   R30 R0 RAM        * Load start address of RAM
        lw   R1  R0 USER       * Load start address of user program
        jalr R1                * Jump to user program

ende:   beqz R0 ende
```

After 8 tests are executed by test program, the 9th test reads and writes to the whole user space to test the memory (which is something one would not like to happen if a user's program is stored in that area). Therefore, before the 9th test starts, the value stored in the address 0x00800000 is compared with a special value (given the name "RESA" in the listing above). If the values are equal, then the 9th test is skipped and a jump to address 0x00800001 is performed. Otherwise, the 9th test takes place.

4.1 Executing a user's program after the test program

This mechanism can be used by you to start your user program only after the 8 tests are executed. To use this mechanism your program should start like the following program:

```
*--------------------------------
* ADD.S
*--------------------------------
        pc= 0x800000           * RAM-Anfang

        dc 0x5245534           * "RESA"
        beqz R0 anfang         * erster Befehl

a:      dc 10                  * 0x800002
b:      dc 20                  * 0x800003
erg:    ds 1

anfang: lw   R1  R30  a
        lw   R2  R30  b
        add  R1  R1   R2
        sw   R1  R30  erg      * erg = a + b

ende:   beqz R0  ende
```

Note that (a) In address 0x00800000 the constant corresponding to the "RESA" special value is stored. (b) The jump to address 0x00800001 is implemented in the test program by a jalr instruction (it could be also implemented by a jr instruction). The register R1 is set to hold the value 0x00800001 (which can not be stored by a 16-bit immediate constant) using a variable that is initialized to this value in the test program. (c) The register R30 is set to hold the value 0x00800000 also by using a variable that is initialized to this value in the test program. The R30 register can then be used by the user's program as the offset to compute addresses so that the user's program variables are accessible (e.g. load and store instructions).

The problem with this having to run 8 tests before the user's program is executed is that it a failure in one of the 8 tests (due to an error in the DLX's functionality) might cause an endless loop, in which case the user's program is not executed at all.

4.2 Executing a user's program without the test program

We avoid running the test program by mapping the logical address 0x00000000 to the physical address 0x00800000, as described in the section on translating addresses.

5 The ID register

The 32-bit ID register is used to check whether the design is yours. The ID register is a read-only register and is accessed by the monitor slave using the an address specified by the RESA control's configuration file. The label `PROC_ID` is reserved for the ID register.

Homework

1. Design DLX test procedures for your DLX design. Describe the assumptions made by the procedure, the tested components, and the test method.

2. Design an simplified DLX. Apply your test and the EPROM test program to validate the correctness of your design.

3. Write a user program that calculates prime numbers as follows. It reads two integers a and b from addresses 0x00800002 and 0x00800003. It computes the prime numbers in the interval $[a, b]$. When the program ends, an array of the prime numbers in the interval $[a, b]$ is stored starting at address 0x00810000. The 32 least significant bits of the sum of the prime numbers in the range $[a, b]$ are stored address 0x0080ffff. Hint: use the Sieve Algorithm of Eratosthenes.

   Execute your program on your DLX design and on the DLX design provided to you.

5.1 Remarks about the RESA control software

The RESA control software can be used to display waveforms in one of two formats: (a) Graphical: The characters "_" and "–" are used to depict low and high values. This option is selected in the configuration file using the "w" type. (b) Bits: The bits "0" and "1" are used to depict low and high values. This option is selected in the configuration file using the "n" type.

5.2 The Logic-Analyzer library macro

A Logic-Analyzer has been added to the "lablib1" library. The inputs of the Logic-Analyzer are:

1. `clk`: clock.

2. `step_en`: the step enable signal from the I/O control logic.

3. `in_init`: indicates whether the control of the DLX was in the "init" state in the beginning of this clock period. Note that this is a combinational signal.

4. `LA_in[31:0]`: This is the input bus which carries the signals to be sampled. Note that the selection of which 32 bits to sample should be done before the logic-analyzer.

5. `ADDR[4:0]`: These are the 5 bits for the address (to be taken from the command register).

The outputs of the Logic-Analyzer are:

1. `LA_out[31:0]`: This is the output bus which carries the read signals from the RAM.

2. `MAXADDR[4:0]`: This is the value of the internal counter. This signals tell us how many cycles have been sampled.

A The DLX test program

Enclosed is the listing of the DLX test program that is stored in the EPROM in the main memory starting at address 0x00000000.

```
*-------------------------------------------------------------------*
* Example of a test program for the DLX
*-------------------------------------------------------------------*
*
* Test0: Fetch / Addresses / Jump Register
* Test1: Databus of RESA-Bus
* Test2: ALU: AddI
* Test3: Shifter
* Test4: Registerfile
* Test5: ALU: Add
* Test6: ALU: Or, And, Xor, TestSet
* Test7: Branch / JALR
* Test8: RAM Test (optional)
*
*-------------------------------------------------------------------*

*-------------------------------------------------------------------*
* Test0: Fetch / Addresses / Jump Register
*-------------------------------------------------------------------*

0x00000000: 0xC0000000      test0: DC 0xC0000000 * Special-Nop
0x00000001: 0x2C000000             NOP                           * == addi R0 R0 0x0
0x00000002: 0x2C000000             NOP
0x00000003: 0x2C000000             NOP
0x00000004: 0x2C000000             NOP
0x00000005: 0x2C000000             NOP
0x00000006: 0x2C000000             NOP
0x00000007: 0x2C000000             NOP
0x00000008: 0x2C000000             NOP
0x00000009: 0x2C000000             NOP
0x0000000A: 0x2C000000             NOP
0x0000000B: 0x2C000000             NOP
0x0000000C: 0x2C000000             NOP
0x0000000D: 0x2C000000             NOP
0x0000000E: 0x2C000000             NOP
0x0000000F: 0x2C000000             NOP
0x00000010: 0x8C010049             LW R1 R0 adr1
0x00000011: 0x58200000             JR R1
0x00000012: 0x1000FFFF     error0: BEQZ R0 error0
*-------------------------------------------------------------------*
```



```
* Variables
*-------------------------------------------------------------------*

0x00000020: 0x00000001       bit0: DC 0x00000001
0x00000021: 0x00000002       bit1: DC 0x00000002
0x00000022: 0x00000004       bit2: DC 0x00000004
0x00000023: 0x00000008       bit3: DC 0x00000008
0x00000024: 0x00000010       bit4: DC 0x00000010
0x00000025: 0x00000020       bit5: DC 0x00000020
0x00000026: 0x00000040       bit6: DC 0x00000040
0x00000027: 0x00000080       bit7: DC 0x00000080
0x00000028: 0x00000100       bit8: DC 0x00000100
0x00000029: 0x00000200       bit9: DC 0x00000200
0x0000002A: 0x00000400      bit10: DC 0x00000400
0x0000002B: 0x00000800      bit11: DC 0x00000800
0x0000002C: 0x00001000      bit12: DC 0x00001000
0x0000002D: 0x00002000      bit13: DC 0x00002000
0x0000002E: 0x00004000      bit14: DC 0x00004000
0x0000002F: 0x00008000      bit15: DC 0x00008000
0x00000030: 0x00010000      bit16: DC 0x00010000
0x00000031: 0x00020000      bit17: DC 0x00020000
0x00000032: 0x00040000      bit18: DC 0x00040000
0x00000033: 0x00080000      bit19: DC 0x00080000
0x00000034: 0x00100000      bit20: DC 0x00100000
0x00000035: 0x00200000      bit21: DC 0x00200000
0x00000036: 0x00400000      bit22: DC 0x00400000
0x00000037: 0x00800000      bit23: DC 0x00800000
0x00000038: 0x01000000      bit24: DC 0x01000000
0x00000039: 0x02000000      bit25: DC 0x02000000
0x0000003A: 0x04000000      bit26: DC 0x04000000
0x0000003B: 0x08000000      bit27: DC 0x08000000
0x0000003C: 0x10000000      bit28: DC 0x10000000
0x0000003D: 0x20000000      bit29: DC 0x20000000
0x0000003E: 0x40000000      bit30: DC 0x40000000
0x0000003F: 0x80000000      bit31: DC 0x80000000
0x00000040: 0x00000000       null: DC 0x00000000
0x00000041: 0x00000000      mask0: DC 0x00000000
0x00000042: 0x000000FF      mask8: DC 0x000000FF
0x00000043: 0x0000FFFF     mask16: DC 0x0000FFFF
0x00000044: 0x00FFFFFF     mask24: DC 0x00FFFFFF
0x00000045: 0xFFFFFFFF     mask32: DC 0xFFFFFFFF
0x00000046: 0xA5A5A5A5     maska5: DC 0xA5A5A5A5
0x00000047: 0x5A5A5A5A     mask5a: DC 0x5A5A5A5A
0x00000048: 0x00000000       adr0: DC 0x00000000
0x00000049: 0x00000100       adr1: DC 0x00000100
0x0000004A: 0x00000200       adr2: DC 0x00000200
0x0000004B: 0x00000300       adr3: DC 0x00000300
0x0000004C: 0x00000400       adr4: DC 0x00000400
0x0000004D: 0x00000500       adr5: DC 0x00000500
0x0000004E: 0x00000600       adr6: DC 0x00000600
0x0000004F: 0x00000700       adr7: DC 0x00000700
0x00000050: 0x00000800       adr8: DC 0x00000800
0x00000051: 0x52455341       RESA: DC 0x52455341 * "RESA"
0x00000052: 0x00800001       USER: DC 0x00800001 * Start addresse of user program
0x00000053: 0x00800000        RAM: DC 0x00800000
0x00000054: 0x00800000     ramadr: DC 0x00800000
0x00000055: 0x00900000     ramend: DC 0x00900000
*-------------------------------------------------------------------*
* Test1: Databus of RESA-Bus
*-------------------------------------------------------------------*

0x00000100: 0xC0000001      test1: DC 0xC0000001 * Special-NOP
0x00000101: 0x8C010020             LW R1 R0 bit0
0x00000102: 0x8C010021             LW R1 R0 bit1
0x00000103: 0x8C010022             LW R1 R0 bit2
0x00000104: 0x8C010023             LW R1 R0 bit3
0x00000105: 0x8C010024             LW R1 R0 bit4
0x00000106: 0x8C010025             LW R1 R0 bit5
0x00000107: 0x8C010026             LW R1 R0 bit6
0x00000108: 0x8C010027             LW R1 R0 bit7
0x00000109: 0x8C010028             LW R1 R0 bit8
0x0000010A: 0x8C010029             LW R1 R0 bit9
```

```
0x0000010B: 0x8C01002A            LW R1 R0 bit10
0x0000010C: 0x8C01002B            LW R1 R0 bit11
0x0000010D: 0x8C01002C            LW R1 R0 bit12
0x0000010E: 0x8C01002D            LW R1 R0 bit13
0x0000010F: 0x8C01002E            LW R1 R0 bit14
0x00000110: 0x8C01002F            LW R1 R0 bit15
0x00000111: 0x8C010030            LW R1 R0 bit16
0x00000112: 0x8C010031            LW R1 R0 bit17
0x00000113: 0x8C010032            LW R1 R0 bit18
0x00000114: 0x8C010033            LW R1 R0 bit19
0x00000115: 0x8C010034            LW R1 R0 bit20
0x00000116: 0x8C010035            LW R1 R0 bit21
0x00000117: 0x8C010036            LW R1 R0 bit22
0x00000118: 0x8C010037            LW R1 R0 bit23
0x00000119: 0x8C010038            LW R1 R0 bit24
0x0000011A: 0x8C010039            LW R1 R0 bit25
0x0000011B: 0x8C01003A            LW R1 R0 bit26
0x0000011C: 0x8C01003B            LW R1 R0 bit27
0x0000011D: 0x8C01003C            LW R1 R0 bit28
0x0000011E: 0x8C01003D            LW R1 R0 bit29
0x0000011F: 0x8C01003E            LW R1 R0 bit30
0x00000120: 0x8C01003F            LW R1 R0 bit31
0x00000121: 0x8C01004A            LW R1 R0 adr2
0x00000122: 0x58200000            JR R1
*-------------------------------------------------------------------*
* Test2: ALU: AddI
*-------------------------------------------------------------------*

0x00000200: 0xC0000002      test2: DC 0xC0000002 * Special-NOP
0x00000201: 0x2C010001            ADDI R1 R0 0x0001
0x00000202: 0x2C010002            ADDI R1 R0 0x0002
0x00000203: 0x2C010004            ADDI R1 R0 0x0004
0x00000204: 0x2C010008            ADDI R1 R0 0x0008
0x00000205: 0x2C010010            ADDI R1 R0 0x0010
0x00000206: 0x2C010020            ADDI R1 R0 0x0020
0x00000207: 0x2C010040            ADDI R1 R0 0x0040
0x00000208: 0x2C010080            ADDI R1 R0 0x0080
0x00000209: 0x2C010100            ADDI R1 R0 0x0100
0x0000020A: 0x2C010200            ADDI R1 R0 0x0200
0x0000020B: 0x2C010400            ADDI R1 R0 0x0400
0x0000020C: 0x2C010800            ADDI R1 R0 0x0800
0x0000020D: 0x2C011000            ADDI R1 R0 0x1000
0x0000020E: 0x2C012000            ADDI R1 R0 0x2000
0x0000020F: 0x2C014000            ADDI R1 R0 0x4000
0x00000210: 0x2C018000            ADDI R1 R0 0x8000
0x00000211: 0x8C01004B            LW R1 R0 adr3
0x00000212: 0x58200000            JR R1
*-------------------------------------------------------------------*
* Test3: Shifter
*-------------------------------------------------------------------*

0x00000300: 0xC0000003      test3: DC 0xC0000003 * Special-NOP
0x00000301: 0x8C010020            LW R1 R0 bit0
0x00000302: 0x00200840            SLLI R1 R1
0x00000303: 0x00200840            SLLI R1 R1
0x00000304: 0x00200840            SLLI R1 R1
0x00000305: 0x00200840            SLLI R1 R1
0x00000306: 0x00200840            SLLI R1 R1
0x00000307: 0x00200840            SLLI R1 R1
0x00000308: 0x00200840            SLLI R1 R1
0x00000309: 0x00200840            SLLI R1 R1
0x0000030A: 0x00200840            SLLI R1 R1
0x0000030B: 0x00200840            SLLI R1 R1
0x0000030C: 0x00200840            SLLI R1 R1
0x0000030D: 0x00200840            SLLI R1 R1
0x0000030E: 0x00200840            SLLI R1 R1
0x0000030F: 0x00200840            SLLI R1 R1
0x00000310: 0x00200840            SLLI R1 R1
0x00000311: 0x00200840            SLLI R1 R1
0x00000312: 0x00200840            SLLI R1 R1
0x00000313: 0x00200840            SLLI R1 R1
0x00000314: 0x00200840            SLLI R1 R1
```



```
0x00000315: 0x00200840            SLLI R1 R1
0x00000316: 0x00200840            SLLI R1 R1
0x00000317: 0x00200840            SLLI R1 R1
0x00000318: 0x00200840            SLLI R1 R1
0x00000319: 0x00200840            SLLI R1 R1
0x0000031A: 0x00200840            SLLI R1 R1
0x0000031B: 0x00200840            SLLI R1 R1
0x0000031C: 0x00200840            SLLI R1 R1
0x0000031D: 0x00200840            SLLI R1 R1
0x0000031E: 0x00200840            SLLI R1 R1
0x0000031F: 0x00200840            SLLI R1 R1
0x00000320: 0x00200840            SLLI R1 R1
0x00000321: 0x00200840            SLLI R1 R1
0x00000322: 0x8C01003F            LW R1 R0 bit31
0x00000323: 0x00200842            SRLI R1 R1
0x00000324: 0x00200842            SRLI R1 R1
0x00000325: 0x00200842            SRLI R1 R1
0x00000326: 0x00200842            SRLI R1 R1
0x00000327: 0x00200842            SRLI R1 R1
0x00000328: 0x00200842            SRLI R1 R1
0x00000329: 0x00200842            SRLI R1 R1
0x0000032A: 0x00200842            SRLI R1 R1
0x0000032B: 0x00200842            SRLI R1 R1
0x0000032C: 0x00200842            SRLI R1 R1
0x0000032D: 0x00200842            SRLI R1 R1
0x0000032E: 0x00200842            SRLI R1 R1
0x0000032F: 0x00200842            SRLI R1 R1
0x00000330: 0x00200842            SRLI R1 R1
0x00000331: 0x00200842            SRLI R1 R1
0x00000332: 0x00200842            SRLI R1 R1
0x00000333: 0x00200842            SRLI R1 R1
0x00000334: 0x00200842            SRLI R1 R1
0x00000335: 0x00200842            SRLI R1 R1
0x00000336: 0x00200842            SRLI R1 R1
0x00000337: 0x00200842            SRLI R1 R1
0x00000338: 0x00200842            SRLI R1 R1
0x00000339: 0x00200842            SRLI R1 R1
0x0000033A: 0x00200842            SRLI R1 R1
0x0000033B: 0x00200842            SRLI R1 R1
0x0000033C: 0x00200842            SRLI R1 R1
0x0000033D: 0x00200842            SRLI R1 R1
0x0000033E: 0x00200842            SRLI R1 R1
0x0000033F: 0x00200842            SRLI R1 R1
0x00000340: 0x00200842            SRLI R1 R1
0x00000341: 0x00200842            SRLI R1 R1
0x00000342: 0x00200842            SRLI R1 R1
0x00000343: 0x8C01004C            LW R1 R0 adr4
0x00000344: 0x58200000            JR R1
*-------------------------------------------------------------------*
* Test4: Registerfile
*-------------------------------------------------------------------*

0x00000400: 0xC0000004      test4: DC 0xC0000004 * Special-NOP
0x00000401: 0x2C000000            ADDI R0 R0 0
0x00000402: 0x2C010001            ADDI R1 R0 1
0x00000403: 0x2C020002            ADDI R2 R0 2
0x00000404: 0x2C030003            ADDI R3 R0 3
0x00000405: 0x2C040004            ADDI R4 R0 4
0x00000406: 0x2C050005            ADDI R5 R0 5
0x00000407: 0x2C060006            ADDI R6 R0 6
0x00000408: 0x2C070007            ADDI R7 R0 7
0x00000409: 0x2C080008            ADDI R8 R0 8
0x0000040A: 0x2C090009            ADDI R9 R0 9
0x0000040B: 0x2C0A000A            ADDI R10 R0 10
0x0000040C: 0x2C0B000B            ADDI R11 R0 11
0x0000040D: 0x2C0C000C            ADDI R12 R0 12
0x0000040E: 0x2C0D000D            ADDI R13 R0 13
0x0000040F: 0x2C0E000E            ADDI R14 R0 14
0x00000410: 0x2C0F000F            ADDI R15 R0 15
0x00000411: 0x2C100010            ADDI R16 R0 16
0x00000412: 0x2C110011            ADDI R17 R0 17
0x00000413: 0x2C120012            ADDI R18 R0 18
```

```
0x00000414: 0x2C130013            ADDI R19 R0 19
0x00000415: 0x2C140014            ADDI R20 R0 20
0x00000416: 0x2C150015            ADDI R21 R0 21
0x00000417: 0x2C160016            ADDI R22 R0 22
0x00000418: 0x2C170017            ADDI R23 R0 23
0x00000419: 0x2C180018            ADDI R24 R0 24
0x0000041A: 0x2C190019            ADDI R25 R0 25
0x0000041B: 0x2C1A001A            ADDI R26 R0 26
0x0000041C: 0x2C1B001B            ADDI R27 R0 27
0x0000041D: 0x2C1C001C            ADDI R28 R0 28
0x0000041E: 0x2C1D001D            ADDI R29 R0 29
0x0000041F: 0x2C1E001E            ADDI R30 R0 30
0x00000420: 0x2C1F001F            ADDI R31 R0 31
0x00000421: 0x2C000000            ADDI R0 R0 0x0
0x00000422: 0x2C210000            ADDI R1 R1 0x0
0x00000423: 0x2C420000            ADDI R2 R2 0x0
0x00000424: 0x2C630000            ADDI R3 R3 0x0
0x00000425: 0x2C840000            ADDI R4 R4 0x0
0x00000426: 0x2CA50000            ADDI R5 R5 0x0
0x00000427: 0x2CC60000            ADDI R6 R6 0x0
0x00000428: 0x2CE70000            ADDI R7 R7 0x0
0x00000429: 0x2D080000            ADDI R8 R8 0x0
0x0000042A: 0x2D290000            ADDI R9 R9 0x0
0x0000042B: 0x2D4A0000            ADDI R10 R10 0x0
0x0000042C: 0x2D6B0000            ADDI R11 R11 0x0
0x0000042D: 0x2D8C0000            ADDI R12 R12 0x0
0x0000042E: 0x2DAD0000            ADDI R13 R13 0x0
0x0000042F: 0x2DCE0000            ADDI R14 R14 0x0
0x00000430: 0x2DEF0000            ADDI R15 R15 0x0
0x00000431: 0x2E100000            ADDI R16 R16 0x0
0x00000432: 0x2E310000            ADDI R17 R17 0x0
0x00000433: 0x2E520000            ADDI R18 R18 0x0
0x00000434: 0x2E730000            ADDI R19 R19 0x0
0x00000435: 0x2E940000            ADDI R20 R20 0x0
0x00000436: 0x2EB50000            ADDI R21 R21 0x0
0x00000437: 0x2ED60000            ADDI R22 R22 0x0
0x00000438: 0x2EF70000            ADDI R23 R23 0x0
0x00000439: 0x2F180000            ADDI R24 R24 0x0
0x0000043A: 0x2F390000            ADDI R25 R25 0x0
0x0000043B: 0x2F5A0000            ADDI R26 R26 0x0
0x0000043C: 0x2F7B0000            ADDI R27 R27 0x0
0x0000043D: 0x2F9C0000            ADDI R28 R28 0x0
0x0000043E: 0x2FBD0000            ADDI R29 R29 0x0
0x0000043F: 0x2FDE0000            ADDI R30 R30 0x0
0x00000440: 0x2FFF0000            ADDI R31 R31 0x0
0x00000441: 0x8C01004D            LW R1 R0 adr5
0x00000442: 0x58200000            JR R1
*-------------------------------------------------------------------*
* Test5: ALU: Add
*-------------------------------------------------------------------*

0x00000500: 0xC0000005      test5: DC 0xC0000005 * Special-NOP
0x00000501: 0x2C020000            ADDI R2 R0 0x0
0x00000502: 0x8C010020            LW R1 R0 bit0
0x00000503: 0x00411023            ADD R2 R2 R1
0x00000504: 0x8C010021            LW R1 R0 bit1
0x00000505: 0x00411023            ADD R2 R2 R1
0x00000506: 0x8C010022            LW R1 R0 bit2
0x00000507: 0x00411023            ADD R2 R2 R1
0x00000508: 0x8C010023            LW R1 R0 bit3
0x00000509: 0x00411023            ADD R2 R2 R1
0x0000050A: 0x8C010024            LW R1 R0 bit4
0x0000050B: 0x00411023            ADD R2 R2 R1
0x0000050C: 0x8C010025            LW R1 R0 bit5
0x0000050D: 0x00411023            ADD R2 R2 R1
0x0000050E: 0x8C010026            LW R1 R0 bit6
0x0000050F: 0x00411023            ADD R2 R2 R1
0x00000510: 0x8C010027            LW R1 R0 bit7
0x00000511: 0x00411023            ADD R2 R2 R1
0x00000512: 0x8C010028            LW R1 R0 bit8
0x00000513: 0x00411023            ADD R2 R2 R1
0x00000514: 0x8C010029            LW R1 R0 bit9
```



```
0x00000515: 0x00411023            ADD R2 R2 R1
0x00000516: 0x8C01002A            LW R1 R0 bit10
0x00000517: 0x00411023            ADD R2 R2 R1
0x00000518: 0x8C01002B            LW R1 R0 bit11
0x00000519: 0x00411023            ADD R2 R2 R1
0x0000051A: 0x8C01002C            LW R1 R0 bit12
0x0000051B: 0x00411023            ADD R2 R2 R1
0x0000051C: 0x8C01002D            LW R1 R0 bit13
0x0000051D: 0x00411023            ADD R2 R2 R1
0x0000051E: 0x8C01002E            LW R1 R0 bit14
0x0000051F: 0x00411023            ADD R2 R2 R1
0x00000520: 0x8C01002F            LW R1 R0 bit15
0x00000521: 0x00411023            ADD R2 R2 R1
0x00000522: 0x8C010030            LW R1 R0 bit16
0x00000523: 0x00411023            ADD R2 R2 R1
0x00000524: 0x8C010031            LW R1 R0 bit17
0x00000525: 0x00411023            ADD R2 R2 R1
0x00000526: 0x8C010032            LW R1 R0 bit18
0x00000527: 0x00411023            ADD R2 R2 R1
0x00000528: 0x8C010033            LW R1 R0 bit19
0x00000529: 0x00411023            ADD R2 R2 R1
0x0000052A: 0x8C010034            LW R1 R0 bit20
0x0000052B: 0x00411023            ADD R2 R2 R1
0x0000052C: 0x8C010035            LW R1 R0 bit21
0x0000052D: 0x00411023            ADD R2 R2 R1
0x0000052E: 0x8C010036            LW R1 R0 bit22
0x0000052F: 0x00411023            ADD R2 R2 R1
0x00000530: 0x8C010037            LW R1 R0 bit23
0x00000531: 0x00411023            ADD R2 R2 R1
0x00000532: 0x8C010038            LW R1 R0 bit24
0x00000533: 0x00411023            ADD R2 R2 R1
0x00000534: 0x8C010039            LW R1 R0 bit25
0x00000535: 0x00411023            ADD R2 R2 R1
0x00000536: 0x8C01003A            LW R1 R0 bit26
0x00000537: 0x00411023            ADD R2 R2 R1
0x00000538: 0x8C01003B            LW R1 R0 bit27
0x00000539: 0x00411023            ADD R2 R2 R1
0x0000053A: 0x8C01003C            LW R1 R0 bit28
0x0000053B: 0x00411023            ADD R2 R2 R1
0x0000053C: 0x8C01003D            LW R1 R0 bit29
0x0000053D: 0x00411023            ADD R2 R2 R1
0x0000053E: 0x8C01003E            LW R1 R0 bit30
0x0000053F: 0x00411023            ADD R2 R2 R1
0x00000540: 0x8C01003F            LW R1 R0 bit31
0x00000541: 0x00411023            ADD R2 R2 R1
0x00000542: 0x2C420001            ADDI R2 R2 0x1              * Zero in ACC (test carry out)
0x00000543: 0x2C02FFFF            ADDI R2 R0 0xffff
0x00000544: 0x8C010020            LW R1 R0 bit0
0x00000545: 0x00411022            SUB R2 R2 R1
0x00000546: 0x8C010021            LW R1 R0 bit1
0x00000547: 0x00411022            SUB R2 R2 R1
0x00000548: 0x8C010022            LW R1 R0 bit2
0x00000549: 0x00411022            SUB R2 R2 R1
0x0000054A: 0x8C010023            LW R1 R0 bit3
0x0000054B: 0x00411022            SUB R2 R2 R1
0x0000054C: 0x8C010024            LW R1 R0 bit4
0x0000054D: 0x00411022            SUB R2 R2 R1
0x0000054E: 0x8C010025            LW R1 R0 bit5
0x0000054F: 0x00411022            SUB R2 R2 R1
0x00000550: 0x8C010026            LW R1 R0 bit6
0x00000551: 0x00411022            SUB R2 R2 R1
0x00000552: 0x8C010027            LW R1 R0 bit7
0x00000553: 0x00411022            SUB R2 R2 R1
0x00000554: 0x8C010028            LW R1 R0 bit8
0x00000555: 0x00411022            SUB R2 R2 R1
0x00000556: 0x8C010029            LW R1 R0 bit9
0x00000557: 0x00411022            SUB R2 R2 R1
0x00000558: 0x8C01002A            LW R1 R0 bit10
0x00000559: 0x00411022            SUB R2 R2 R1
0x0000055A: 0x8C01002B            LW R1 R0 bit11
0x0000055B: 0x00411022            SUB R2 R2 R1
0x0000055C: 0x8C01002C            LW R1 R0 bit12
```

```
0x0000055D: 0x00411022             SUB R2 R2 R1
0x0000055E: 0x8C01002D             LW R1 R0 bit13
0x0000055F: 0x00411022             SUB R2 R2 R1
0x00000560: 0x8C01002E             LW R1 R0 bit14
0x00000561: 0x00411022             SUB R2 R2 R1
0x00000562: 0x8C01002F             LW R1 R0 bit15
0x00000563: 0x00411022             SUB R2 R2 R1
0x00000564: 0x8C010030             LW R1 R0 bit16
0x00000565: 0x00411022             SUB R2 R2 R1
0x00000566: 0x8C010031             LW R1 R0 bit17
0x00000567: 0x00411022             SUB R2 R2 R1
0x00000568: 0x8C010032             LW R1 R0 bit18
0x00000569: 0x00411022             SUB R2 R2 R1
0x0000056A: 0x8C010033             LW R1 R0 bit19
0x0000056B: 0x00411022             SUB R2 R2 R1
0x0000056C: 0x8C010034             LW R1 R0 bit20
0x0000056D: 0x00411022             SUB R2 R2 R1
0x0000056E: 0x8C010035             LW R1 R0 bit21
0x0000056F: 0x00411022             SUB R2 R2 R1
0x00000570: 0x8C010036             LW R1 R0 bit22
0x00000571: 0x00411022             SUB R2 R2 R1
0x00000572: 0x8C010037             LW R1 R0 bit23
0x00000573: 0x00411022             SUB R2 R2 R1
0x00000574: 0x8C010038             LW R1 R0 bit24
0x00000575: 0x00411022             SUB R2 R2 R1
0x00000576: 0x8C010039             LW R1 R0 bit25
0x00000577: 0x00411022             SUB R2 R2 R1
0x00000578: 0x8C01003A             LW R1 R0 bit26
0x00000579: 0x00411022             SUB R2 R2 R1
0x0000057A: 0x8C01003B             LW R1 R0 bit27
0x0000057B: 0x00411022             SUB R2 R2 R1
0x0000057C: 0x8C01003C             LW R1 R0 bit28
0x0000057D: 0x00411022             SUB R2 R2 R1
0x0000057E: 0x8C01003D             LW R1 R0 bit29
0x0000057F: 0x00411022             SUB R2 R2 R1
0x00000580: 0x8C01003E             LW R1 R0 bit30
0x00000581: 0x00411022             SUB R2 R2 R1
0x00000582: 0x8C01003F             LW R1 R0 bit31
0x00000583: 0x00411022             SUB R2 R2 R1
0x00000584: 0x2C42FFFF             ADDI R2 R2 -1                * -1 in ACC (test carry out)
0x00000585: 0x8C01004E             LW R1 R0 adr6
0x00000586: 0x58200000             JR R1
*----------------------------------------------------------------*
* Test6: ALU: Or, And, Xor, TestSet
*----------------------------------------------------------------*

0x00000600: 0xC0000006      test6: DC 0xC0000006 * Special-NOP
0x00000601: 0x8C020041             LW R2 R0 mask0
0x00000602: 0x8C010041             LW R1 R0 mask0
0x00000603: 0x00411025             OR R2 R2 R1                  * acc -> 0x00000000
0x00000604: 0x8C010042             LW R1 R0 mask8
0x00000605: 0x00411025             OR R2 R2 R1
0x00000606: 0x8C010043             LW R1 R0 mask16
0x00000607: 0x00411025             OR R2 R2 R1
0x00000608: 0x8C010044             LW R1 R0 mask24
0x00000609: 0x00411025             OR R2 R2 R1
0x0000060A: 0x8C010045             LW R1 R0 mask32
0x0000060B: 0x00411025             OR R2 R2 R1                  * acc -> 0xffffffff
0x0000060C: 0x8C020045             LW R2 R0 mask32
0x0000060D: 0x8C010045             LW R1 R0 mask32
0x0000060E: 0x00411026             AND R2 R2 R1                 * acc -> 0xffffffff
0x0000060F: 0x8C010044             LW R1 R0 mask24
0x00000610: 0x00411026             AND R2 R2 R1
0x00000611: 0x8C010043             LW R1 R0 mask16
0x00000612: 0x00411026             AND R2 R2 R1
0x00000613: 0x8C010042             LW R1 R0 mask8
0x00000614: 0x00411026             AND R2 R2 R1
0x00000615: 0x8C010041             LW R1 R0 mask0
0x00000616: 0x00411026             AND R2 R2 R1                 * acc -> 0x00000000
0x00000617: 0x8C020046             LW R2 R0 maska5
0x00000618: 0x8C010046             LW R1 R0 maska5
0x00000619: 0x00411024             XOR R2 R2 R1                 * acc -> 0x00000000
```



```
0x0000061A: 0x8C020047             LW R2 R0 mask5a
0x0000061B: 0x8C010047             LW R1 R0 mask5a
0x0000061C: 0x00411024             XOR R2 R2 R1                 * acc -> 0x00000000
0x0000061D: 0x8C020046             LW R2 R0 maska5              * acc -> 0xa5a5a5a5
0x0000061E: 0x8C010047             LW R1 R0 mask5a
0x0000061F: 0x00411024             XOR R2 R2 R1                 * acc -> 0xffffffff
0x00000620: 0x8C010046             LW R1 R0 maska5
0x00000621: 0x00411024             XOR R2 R2 R1                 * acc -> 0x5a5a5a5a
0x00000622: 0x8C010047             LW R1 R0 mask5a
0x00000623: 0x00411024             XOR R2 R2 R1                 * acc -> 0x00000000
0x00000624: 0x2C010000             ADDI R1 R0 0x0
0x00000625: 0x64220000             SGTI R2 R1 0
0x00000626: 0x68220000             SEQI R2 R1 0
0x00000627: 0x6C220000             SGEI R2 R1 0
0x00000628: 0x70220000             SLTI R2 R1 0
0x00000629: 0x74220000             SNEI R2 R1 0
0x0000062A: 0x78220000             SLEI R2 R1 0
0x0000062B: 0x64220001             SGTI R2 R1 1
0x0000062C: 0x68220001             SEQI R2 R1 1
0x0000062D: 0x6C220001             SGEI R2 R1 1
0x0000062E: 0x70220001             SLTI R2 R1 1
0x0000062F: 0x74220001             SNEI R2 R1 1
0x00000630: 0x78220001             SLEI R2 R1 1
0x00000631: 0x6422FFFF             SGTI R2 R1 -1
0x00000632: 0x6822FFFF             SEQI R2 R1 -1
0x00000633: 0x6C22FFFF             SGEI R2 R1 -1
0x00000634: 0x7022FFFF             SLTI R2 R1 -1
0x00000635: 0x7422FFFF             SNEI R2 R1 -1
0x00000636: 0x7822FFFF             SLEI R2 R1 -1
0x00000637: 0x2C010001             ADDI R1 R0 1
0x00000638: 0x64220000             SGTI R2 R1 0x0
0x00000639: 0x68220000             SEQI R2 R1 0x0
0x0000063A: 0x6C220000             SGEI R2 R1 0x0
0x0000063B: 0x70220000             SLTI R2 R1 0x0
0x0000063C: 0x74220000             SNEI R2 R1 0x0
0x0000063D: 0x78220000             SLEI R2 R1 0x0
0x0000063E: 0x2C01FFFF             ADDI R1 R0 -1
0x0000063F: 0x64220000             SGTI R2 R1 0x0
0x00000640: 0x68220000             SEQI R2 R1 0x0
0x00000641: 0x6C220000             SGEI R2 R1 0x0
0x00000642: 0x70220000             SLTI R2 R1 0x0
0x00000643: 0x74220000             SNEI R2 R1 0x0
0x00000644: 0x78220000             SLEI R2 R1 0x0
0x00000645: 0x8C01004F             LW R1 R0 adr7
0x00000646: 0x58200000             JR R1
*----------------------------------------------------------------*
* Test7: Branch / JALR
*----------------------------------------------------------------*

0x00000700: 0xC0000007      test7: DC 0xC0000007 * Special-NOP
0x00000701: 0x2C010000             ADDI R1 R0 0x0               * Branch-Test
0x00000702: 0x14200014             BNEZ R1 error7               * invalid branch condition
0x00000703: 0x10200003             BEQZ R1 test7b               * branch forward
0x00000704: 0x8C01004F             LW R1 R0 adr7
0x00000705: 0x58200000             JR R1
0x00000706: 0x10200005     test7a: BEQZ R1 test7c               * branch forward
0x00000707: 0x1020FFFE     test7b: BEQZ R1 test7a               * branch backward
0x00000708: 0x2C000000             NOP
0x00000709: 0x2C000000             NOP
0x0000070A: 0x8C01004F             LW R1 R0 adr7
0x0000070B: 0x58200000             JR R1
0x0000070C: 0x2C010001     test7c: ADDI R1 R0 0x1
0x0000070D: 0x10200009             BEQZ R1 error7               * invalid branch condition
0x0000070E: 0x14200003             BNEZ R1 test7e               * branch forward
0x0000070F: 0x8C01004F             LW R1 R0 adr7
0x00000710: 0x58200000             JR R1
0x00000711: 0x14200007     test7d: BNEZ R1 test7f               * branch forward
0x00000712: 0x1420FFFE     test7e: BNEZ R1 test7d               * branch backward
0x00000713: 0x2C000000             NOP
0x00000714: 0x2C000000             NOP
0x00000715: 0x8C01004F             LW R1 R0 adr7
0x00000716: 0x58200000             JR R1
```

```
0x00000717: 0x8C01004F     error7:  LW R1 R0 adr7
0x00000718: 0x58200000              JR R1
0x00000719: 0x2C01071C     test7f:  ADDI R1 R0 jlink              * JALR-Test
0x0000071A: 0x5C200000              JALR R1
0x0000071B: 0x1000FFFF              BEQZ R0 -1                    * Stop, if JALR failed
0x0000071C: 0x2FFF0001      jlink:  ADDI R31 R31 0x1
0x0000071D: 0x03E10824              XOR R1 R31 R1                 * compare R31
0x0000071E: 0x10200001              BEQZ R1 test7g
0x0000071F: 0x1000FFFF              BEQZ R0 -1                    * Stop, if R31 wrongly set
0x00000720: 0x8C010050     test7g:  LW R1 R0 adr8
0x00000721: 0x58200000              JR R1
*-------------------------------------------------------------------*
* Test8: RAM Test (optional)
*-------------------------------------------------------------------*

0x00000800: 0xC0000008      test8: DC 0xC0000008 * Special-NOP
0x00000801: 0x8C1E0053              LW R30 R0 RAM                 * R30 := Address of RAM
0x00000802: 0x8C010051              LW R1 R0 RESA
0x00000803: 0x8FC20000              LW R2 R30 0x0                 * read from M[0x800000]
0x00000804: 0x00411024              XOR R2 R2 R1                  * if (M[0x800000]=="RESA")
0x00000805: 0x10400027              BEQZ R2 start                 *   then jump to user program
0x00000806: 0x14400000              BNEZ R2 ramtst                *   else perform memory test
*-------------------------------------------------------------------*

0x00000807: 0x2C010000     ramtst:  ADDI R1 R0 0x0                * Test of first address of RAM
0x00000808: 0xAFC10000              SW R1 R30 0x0                 * 0x00000000 -> hilf
0x00000809: 0x8FC20000              LW R2 R30 0x0
0x0000080A: 0x00411024              XOR R2 R2 R1
0x0000080B: 0x1440001E              BNEZ R2 error8
0x0000080C: 0x2C01FFFF              ADDI R1 R0 0xffff
0x0000080D: 0xAFC10000              SW R1 R30 0x0                 * 0xffffffff -> hilf
0x0000080E: 0x8FC20000              LW R2 R30 0x0
0x0000080F: 0x00411024              XOR R2 R2 R1
0x00000810: 0x14400019              BNEZ R2 error8
0x00000811: 0x2C010001              ADDI R1 R0 0x1                * Test of each single bit
0x00000812: 0xAFC10000      loop8:  SW R1 R30 0x0                 * Powers of 2 -> hilf
0x00000813: 0x8FC20000              LW R2 R30 0x0
0x00000814: 0x00411024              XOR R2 R2 R1
0x00000815: 0x14400014              BNEZ R2 error8
0x00000816: 0x00200840              SLLI R1 R1
0x00000817: 0x1420FFFA              BNEZ R1 loop8
0x00000818: 0x8C010054     test8a:  LW R1 R0 ramadr               * Test of each single RAM cell
0x00000819: 0x8C020055              LW R2 R0 ramend
0x0000081A: 0xAFC10000     loop8a:  SW R1 R30 0x0
0x0000081B: 0x00221822              SUB R3 R1 R2
0x0000081C: 0x10600003              BEQZ R3 test8b
0x0000081D: 0xAC210000              SW R1 R1 0x0
0x0000081E: 0x2C210001              ADDI R1 R1 0x1
0x0000081F: 0x1000FFFA              BEQZ R0 loop8a
0x00000820: 0x8C010054     test8b:  LW R1 R0 ramadr
0x00000821: 0xAFC10000     loop8b:  SW R1 R30 0x0
0x00000822: 0x00221822              SUB R3 R1 R2
0x00000823: 0x10600007              BEQZ R3 test8c
0x00000824: 0x8C230000              LW R3 R1 0x0                  * Compare memory contents
0x00000825: 0x00611824              XOR R3 R3 R1
0x00000826: 0x14600003              BNEZ R3 error8
0x00000827: 0xAC200000              SW R0 R1 0x0                  * Erase memory
0x00000828: 0x2C210001              ADDI R1 R1 0x1
0x00000829: 0x1000FFF7              BEQZ R0 loop8b
0x0000082A: 0x1000FFFF     error8:  BEQZ R0 error8                * Endless loop
0x0000082B: 0xAFC00000     test8c:  SW R0 R30 0x0                 * Test successful
0x0000082C: 0x1000FFFF      stop8:  BEQZ R0 stop8                 * Endless loop
*-------------------------------------------------------------------*
* Start user program
*-------------------------------------------------------------------*

0x0000082D: 0x8C1E0053      start:  LW R30 R0 RAM                 * Load start address of RAM
0x0000082E: 0x8C010052              LW R1 R0 USER                 * Load start address of user program
0x0000082F: 0x5C200000              JALR R1                       * Jump to user program
0x00000830: 0x1000FFFF       ende:  BEQZ R0 ende
*-------------------------------------------------------------------*
```