University of the Saarland
Department 6.2 - Informatik
Prof. Dr. W.J. Paul

Computer Architecture I - WS 03/04

(due: 17.12.2003)

Exercise 1: (predicates) **(4 points)**

Define the following predicates (boolean functions which take $IR(c)$ as an argument and return true if corresponding condition holds):
(for all predicates p we use shorthand $p(c) = p(IR(c))$)

$rtype(c)$ – $IR(c)$ is an instruction of R-type
$jtype(c)$ – $IR(c)$ is an instruction of J-type
$itype(c)$ – $IR(c)$ is an instruction of I-type
$alui(c)$ – $IR(c)$ is any arithmetic or logic operation with immediate constant
$testi(c)$ – $IR(c)$ is any test operation with immediate constant
$shi(c)$ – $IR(c)$ is any shift operation with immediate constant (shift amount SA)
$alu(c)$ – $IR(c)$ is any arithmetic or logic operation for register operands
$test(c)$ – $IR(c)$ is any test operation for register operands
$sh(c)$ – $IR(c)$ is any shift operation for register operands
$j(c)$ – $IR(c)$ is a jump instruction
$jal(c)$ – $IR(c)$ is a jump and link instruction
$jr(c)$ – $IR(c)$ is a jump register instruction
$jalr(c)$ – $IR(c)$ is a jump and link register instruction
$jump(c)$ – $IR(c)$ is any jump instruction

Exercise 2: (DLX assembler programming) **(7 points)**

Let $n, a_1, \ldots, a_n$ be natural numbers. Write a DLX assembler program which computes the maximum of the numbers $a_1, \ldots, a_n$. At the start of your program n will be stored in register $GPR[1]$ and the a_i in memory cell $M(i-1)$ $(i \in \{1, \ldots, n\})$. Your program should store its result in memory cell $M(n)$. Comment your program in a way that everyone can understand what it should do. Programs without enough comments will get 0 points!!!

Exercise 3: (constant computation) (see notes below) **(7+7 points)**

Specify two variants of a layout computing constant $C_0(IR(c))$.

- Variant 1
  You need to construct a layout *id_simple* providing necessary signals using in the constant computation (such as *jtype*, *shi* etc.).
  Also you need to construct a layout to compute the constant C_0, which takes $IR(C)$ (or necessary bits) and signals from *id_simple* as inputs and returns 32-bit value of a constant corresponding to the current $IR(c)$.
  Specify and analyze both layouts.
  Compute maximal *time* of the output providing the constant for the **combination** of both layouts. In this case you need to include the delay of nets between two circuits (the length of a net between two circuits also includes loads of inputs which are connected to it[1]).

- Variant 2
  In this variant, compute necessary control signals **inside** of the layout computing constant. So, it should take only $IR(c)$ (or necessary bits) as input and provide the same result as in the previous case.
  Specify and analyze this layout.

[1] See *Load and Time* in the internet page under *Layouts*

University of the Saarland

Department 6.2 - Informatik

Prof. Dr. W.J. Paul

Computer Architecture I - WS 03/04

(due: 17.12.2003)

Exercise 4: (different layouts) (see notes below) **(10 points)**

This exercise contains 3 tasks, one task for one exercise group. In each case you need to specify and analyze a layout which will be used in ALU. Give the fastest layout you can.

- **Group 1** (Mo. 16-18)
  Specify a layout for Comparator (see MP00[2] Fig. 3.8). It takes 32-bit input value from AU (difference between two numbers to be compared), 1-bit flag *neg* and 3-bit code of the operation to be done (Table 3.5) and returns 1-bit result of computation.
- **Group 2** (Wed. 14-16)
  Specify a layout for n-bit conditional sum incrementer (see MP00 Fig. 2.15), which takes inputs $a[n-1:0]$ and c_{in}, and returns $s[n:0]$ such that $\langle s\rangle = \langle a\rangle + c_{in}$. Please, pay attention to sizes of incrementers used in recursive construction !!!
- **Group 3** (Fr. 16-18)
  Specify a layout for Logic Unit (see MP00 Fig. 3.9). It takes two 32-bit input values and 2-bit code of the operation to be done (Table 3.6) and it returns 32-bit result of computation.

Notes for constructing layouts:

- Specification of layouts for gates is given in the internet page under *Layouts* (see *Basic elements*).
- Basic layouts (for gates) cannot be modified.
- Locations of inputs/outputs for gates are specified precisely, use them in your computations.
- Consider layouts only for $\delta = 1/3$ and $\nu = 1/3$
- Specifying and analysis means that for each layout you need to give:
  - Clear and detailed drawing of the layout. Pay attention to relative sizes of gates (or better draw in scale).
  - Specify input and output locations, i.e. show in the drawing and compute **precise** distances between inputs and outputs and their locations according to the edge of the layout.
  - Compute height and width of the layout you have specified.
  - Compute *load* for all inputs.
  - Compute maximal *time* in the layout.

[2] Müller, S.M. and Paul, W.J. *Computer Architecture, Complexity and Correctness* Springer Verlag ISBN 3-540-67481-0