

SOLUTIONS
EE4304: Computer Architecture
Mid-Term Exam, October 21, 2009
Prof. V.G. Oklobdzija

| | |
|-----------------------|---------------------------|
| Name: | Score: / 100 |
| ID: | |

*Note: Exam is CLOSED-BOOK EXAM.
100 pints total (points marked at each question)*

1. (10p) Circle the items that are part of the definition of **computer architecture**, cross the ones that are not a part of **computer architecture** definition:

(a) instruction set

~~(b) clock speed~~

(c) addressing modes

(d) register file

~~(e) pipeline organization~~

(f) number of floating-point registers

~~(g) number and type of floating point execution units~~

(h) floating-point formats

~~(i) memory size~~

2. (5p) Who is credited with the invention of:

(a) the first stored-program computer ?

EDVAC - Von Neuman

(b) the first electronic computer ?

Eckert & Mauchly

(c) the first computer ?

Konrad Zuse

3. (5p) Does a compiler need to be written in assembly language? Why? (Explain your answer)

No. Because the compiler is just a parser, which checks the rules for the code to be executed.

4. (10p) What is a decimal value of a single-precision floating-point number:

| S | Exp | Mantissa |
|---|---------|-------------------------|
| 0 | 0000011 | 00010010010000000000000 |

Representation format is: $(-1)^S \times 2^{(Exp-127)} \times$ (value of the fraction)

$$1 \times 2^{-124} \times (1 + 2^{-4} + 2^{-7} + 2^{-10}) = 5.0372 \times 10^{-38}$$

5. (20pts) Compare 0,1,2, and 3 address machines by

(a) writing a program to compute:

$$X = (A+BxC) / (D-ExF-GxH) \dots\dots\dots (1)$$

for each of the four types of instruction sets. The available instructions are:

| 0-address | 1-address | 2-address | 3-address |
|------------------|------------------|------------------|------------------|
| Push M | Load ACC | Mov (R1 <- R2) | Mov (R1<-R2) |
| Pop M | Store ACC | | |
| Add | Add ACC | Add (R1<-R1+R2) | Add (R1<-R2+R3) |
| Sub | Sub ACC | Sub (R1<-R1-R2) | Sub (R1<-R2-R3) |
| Mul | Mul ACC | Mul (R1<-R1XR2) | Mul (R1<-R2xR3) |
| Div | Div ACC | Div (R1<-R1/R2) | Div (R1<-R2/R3) |

(b) assuming that the 0-address machine instructions are 1-byte of length, 1-address: 1.5-bytes, 2-address: 2-bytes and 3-address: 4-bytes,

List the size of the code computing (1) (in terms of bytes) for each machine.

(c) assuming that each load and store deals with a 4-byte data, list the total number of bytes moved between CPU and memory.

a)

| 0-address | 1-address | 2-address | 3-address |
|------------------|------------------|------------------|------------------|
| Push A | Load E | Mov R1 B | Mul R1 B C |
| Push B | Mul F//EF | Mul R1 C | Add R1 R1 A |
| Push C | Store R1 | Add R1 A//a+bc | Mul R2 E F |
| Mul | Load G | Mov R2 E | Mul R3 G H |
| Add //A+BC | Mul H //GH | Mul R2 F | Add R2 R2 R3 |
| Push D | store R2 | Mov R3 G | Sub R2 D R2 |
| Push E | Load D | Mul R3 H | Div Res R1 R2 |
| Push F | sub R1 | Mov R4 D | |
| Mul //EF | sub R2 | Sub R4 R2 | |
| Push G | store R3 | sub R4 R3 | |
| Push H | Load B | Div R1 R4 | |
| Mul //GH | Mul C | Mov Res R1 | |
| Add //GH+EF | Add A | | |
| Push D | Div R3 | | |
| Sub //D -(GH+EF) | Store Res | | |
| Div | | | |
| Pop Res | | | |

b) 0-address : 16 instructions = 16 bytes

1-address : 15 instructions = 22.5 bytes, 2-address : 12 instructions = 24 bytes,

3-address : 7 instructions = 28 bytes

c) 0-address : (8 Push + 1 Pop) = 9 transfers = 36 bytes

1-address : (4 Load's + 4 Store's) = 8 transfers = 32 bytes

2-address and 3-address = zero transfers.

5. (25) Given are the values in memory and in the register file listed by the locations shown below

| Memory | |
|--------|------|
| FE20 | 40 |
| FE21 | FE25 |
| FE22 | FE26 |
| FE23 | FE24 |
| FE24 | FE27 |
| FE25 | FE25 |
| FE26 | F2 |
| FE27 | 20 |

| Register File | |
|---------------|------|
| R0 | FE20 |
| R1 | F |
| R2 | FE23 |
| R3 | 2 |
| R4 | FE20 |
| R5 | FE21 |
| R6 | 0 |
| R7 | FA |

Write the value of the registers after each of the following Load operations:

| Addressing Mode | Instruction | Reg. Value |
|-----------------------------------|-----------------|----------------------|
| Direct (Absolute) | Load R4, FE27 | R4 =20 |
| Register Indirect | Load R4, R2 | R4 =FE24 |
| Memory Indirect | Load R4, FE21 | R4 =FE25 |
| Immediate | Load R4, FE20 | R4 =FE20 |
| Base + Index | Load R4, R2, R3 | R4 =FE25 |
| Base + Displacement | Load R4, R5, 3 | R4 =FE27 |
| Base+Index (pre-increment) | Load R4, R2, R3 | R4 =F2 R3 =3 |
| Base+Index (post-increment) | Load R4, R4, R6 | R4 =40 R6 =1 |
| Base+Index (indirect) | Load R4, R5, R6 | R4 =FE25 |
| Register Indirect (pre-increment) | Load R4, R2 | R4 =FE27 R2 =FE24 |

6. (5) Assign priorities to the following exception conditions from most important to least important (give numbers 1-most important):

- (3) system call
- (1) machine check
- (4) disk I/O
- (2) arithmetic overflow

7. (20p) Loop: lw \$5, 4(\$4)
add \$3, \$5, \$4
or \$7, \$3, \$5
add \$2,\$4,\$7
sub \$5, \$7, \$8
and \$1,\$4,\$6
bneqz \$5, loop

- i) In the above code sequence identify the data dependencies and instructions which can cause stalls in the 5 stage MIPS pipeline.

Data dependencies: Reg \$5 between lw to add instructions,
Reg \$3, between or and add instructions,
Reg \$7, or and add instructions,
Reg \$7, Or and sub instructions
Reg \$5, Sub and bneqz instructions

- ii) Rearrange the instructions in the above loop to optimize the code in terms of number of execution cycles, assume that branch test is done in ID stage.

Loop: lw \$5, 4(\$4)
and \$1,\$4,\$6
add \$3, \$5, \$4
or \$7, \$3, \$5
sub \$5, \$7, \$8
addi \$2,\$4,\$7
bneqz \$5, loop