Name: 08/01/2013

CENG 311 COMPUTER ARCHITECTURE
FINAL EXAM

Duration: 1.5 hours

Q1 (15 pts) The below code is an excerpt from a VHDL program. Assume that at some time, signal `a` changes from "00000001" to "00000000". What would be the final value of signal `x`?

```
(1)     signal a: std_logic_vector(7 downto 0);
(2)     signal x: std_logic;
        ...
(3)     process(a)
(4)     variable b: integer :=1;
(5)     begin
(6)             a<="01010101";
(7)             b:=2;
(8)             x<=a(b);
(9)     end process;
```

ANSWER: When signal a changes, the process is triggered and run. There is a signal assignment in line (6), but the assignment is done at the end of the process execution. Therefore, a(b) and x hold the value of 0 (zero).

Q2 (20 pts) We execute following assembly program on a 32-bit processor. We can understand that `func` represents the starting address of a procedure. Therefore it should normally be called by a "`call func`" command. However it is called with "`jmp func`" as seen in line (9), which would normally cause a problem. Nevertheless, this program executes properly.

```
(1)     section .text
(2)             global _start
(3)     func:   mov eax, 2
(4)             ret
(5)     _start:
(6)             mov eax, callf
(7)             add eax, 6
(8)             push eax
(9)     callf:  jmp func
                ...
```

a) How can you comment on the size of "`jmp`" instruction? In other words, is "`jmp`" 1-byte, 2-byte or 4-byte instruction? Explain your answer.
b) What is the return value of function `func`?

ANSWER: "jmp" is a 2-byte instruction. The return address of the procedure must point the next intruction of "callf: jmp func". This return address is constructed as "callf+6" and put onto stack before the jump instruction. Therefore we can understand that "jmp func" takes 6 bytes. If func represents an 32-bit (4 bytes) address, then "jmp" takes 2 bytes.

Q3 (20 pts) The source codes below constitute one executable code (using "gcc sample.s sample.c -o sample -masm=intel -lm"). During run-time of the program, what would be the content of EAX register at point L1? Explain your answer.

```
sample.s:
        ...
main: ...
        push 5
        push 12
        push 16
        call func
        pop ebx
        add eax, ebx
L1:     ... ; eax=?
```

```
sample.c:

#include<math.h>
struct A{
        int x, y;
};
int func(struct A a)
{
 return sqrt(a.x*a.x+a.y*a.y);
}
```

ANSWER: In "sample.s", three integers are pushed into the stack: 5, 12 and 16. Then function func is called. The function takes the last two integers that exist on the stack and calculates the square root of them as 20.The return value 20 is stored in eax. "pop ebx" command pops 16 from the stack. "add eax, ebx" command sums 20 and 16. Thus, eax register finally holds the value of 36.

Q4 (15 pts) You are supposed to develop an Assembler for microprocessor uPabs2 that you designed this semester. Please give a road map for your design. Explain which tools and languages you will use in your development.

ANSWER: Assembler is basically a "compiler" that takes human understandable mnemonics (opcodes) and translates them into raw binary. Our new assembler can have a user interface, an editor but it is not obligatory. It needs to be able to build a symbol table and a have a linker to resolve references. It can be developed using assembly, C, java or other high level languages. If a parser is needed, lex&yacc or any other tool can be utilized.

Q5(15 pts) Assembly commands `mov eax,eax` and `nop` (no-operation) are semantically identical. Which one do you prefer to implement idle time and why?

ANSWER : We prefer nop command to implement idle time since it is expected to be more energy efficient than other commands.

Q6(15 pts) We want to connect three memory chips, which are 2 kbytes, 4 kbytes and 8 kbytes in size, to a microprocessor that has 32 bit address bus. Please design an address decoder circuit (you can use partial address decoding).

ANSWER: If we use partial address decoding, we can directly connect A31 to the enable input of 8k memory. For the other two enable signals we can produce A31' OR A30 and A31' OR A30', using two OR gates.