Examination
Design of Embedded DSP Processors, TSEA26

Date	2009-10-22
Room	T1 and T2
Time	14:00-18:00
Course code	TSEA26
Exam code	TEN 1
Course name	Design of Embedded DSP Processors
Department	ISY, Department of EE
Number of questions	6
Number of pages (including this page)	7
Responsible teacher	Andreas Ehliar
Phone number during the exam time	013-288956
Visiting the exam room	Around 15.00 and 17.00
Course administrator	Ylva Jernling, 013-282648, ylva@isy.liu.se
Permitted equipment	None, besides an English dictionary
Grading	**Points** / **Swedish grade**: 85-100 → 5; 70-84 → 4; 50-69 → 3; 0-49 → U
Other important information:	• Answers can be given in English or Swedish. • Your answers must be brief and easy to understand. Your grade (number of points) will depend partly on how easy it is for us to understand and verify your answer. • A correct but not justified answer will not give full points on the question. • The width of data buses and registers must be specified. The alignment must be specified in all concatenations of signals or buses. • All numbers are in two's complement format unless otherwise specified. • **Don't write any answers on the exam sheet!**

Grading:

Points	Swedish grade
85-100	5
70-84	4
50-69	3
0-49	U

Question 1 - ALU (15p)

Draw a schematic and a partial control signal table for an arithmetic unit with the following inputs:

`A[3:0]`	Operand A (Always signed)
`B[3:0]`	Operand B (Always signed)

It has the following output: `RESULT[3:0]`

It should be capable of the following six operations:

A+B	A−B
ABS(B)	ABS(A−B)
MAX(A,B)	MIN(A,B)

You should minimize the number of adders in this task. Your hardware should support all of these operations, but your control signal table only needs to show the control signal configuration for operation ABS(A−B), MAX(A,B), and MIN(A,B).

Question 2 - MAC (30p)

Implement a MAC unit which is suitable for the two programs listed on the next page. You are allowed to use only one multiplier. You should also minimize the number of adders. You may use as many multiplexers and logic gates as you want (within reason). The functionality of the MAC unit should be sufficient to execute the two listed programs.

- The register file is 16 bit wide. The memory is 16 bit wide. The accumulator should be 40 bits wide. These 40 bits include 8 guard bits.
- It should be possible to execute both programs in at most 40 clock cycles each.
- You must figure out a suitable size for the multiplier and adder by yourself.

a) Select a suitable instruction set for your MAC unit based on the requirements above and the programs on the next page. (You don't have to choose an instruction encoding.) (5p)

b) Draw a hardware schematic of your MAC unit based on the instructions you selected. (You do not have to implement any part of the DSP processor that is located outside the MAC unit, such as the AGU unit for example.) Draw a control signal table where you show the values of all control signals in your MAC unit for each instruction that you selected. (25p)

Pseudo code for program 1

```
ACR0 = 0;
repeat(32)
   ACR0=ACR0+FRACTIONAL_MULTIPLY(DM0[ADDRPTR0++], DM1[ADDRPTR1++]);
endrepeat

ACR0 = SATURATE(ACR0);

R0 = ACR0[31:16];
R1 = ACR0[15:0];
```

Pseudo code for program 2

```
ACR0 = R14; // R14 and R15 is in fractional format
ACR1 = R15;

repeat(32)
   TMP=FRACTIONAL_MULTIPLY(DM0[ADDRPTR0++], DM1[ADDRPTR1++]);
   if(TMP >= 0)
      ACR0 = ACR0 + TMP;
   else
      ACR1 = ACR1 + TMP;
   endif
endrepeat

ACR1 = ABS(ACR1);
R14 = SATURATE(ROUND(ACR0)); // R14 and R15 should be
R15 = SATURATE(ROUND(ACR1)); // in fractional format
```

Question 3 - General Knowledge (15p)

a) Describe the difference between dynamic and static code profiling (2p)

b) You need a large data memory in your DSP processor. What is most important when you optimize for maximum clock frequency and why? (2p)

Option A: Minimizing the delay of combinational logic connected to the inputs of the memory
Option B: Minimizing the delay of combinational logic connected to the outputs of the memory

c) In a pipelined processor you can have data, control, and structural hazards. Explain what a data hazard is. Explain one way to handle data hazards in a DSP core. (3p)

d) You are designing an FIR filter with 13 taps. Both the coefficients and the input samples are in a fractional format. How many guard bits do you need in your MAC unit to handle the worst case? (2p)

e) `A[3:0] = 1011, B[3:0] = 1001`. `A` is signed, `B` is unsigned. What is the signed 8-bit result of an integer multiplication of `A` and `B`? (You need to show the calculations.) (3p)

f) Explain how an integer adder can be used for fractional addition. (1p)

g) When executing integer multiplication and fractional multiplication based on two's complement data without guard bits, for which multiplication do we have to use saturation after the multiplication and why? (2p)

Question 4 - Two's complement (10p)

Your task is to design a 16×16 multiplier which can handle both unsigned integer multiplication and signed integer multiplication. The multiplication unit should be fully combinational.

Inputs:

`A[15:0]`	Operand A
`B[15:0]`	Operand B
`CTRL`	If this signal is 1, an unsigned multiplication should be performed. If it is 0, a signed multiplication should be performed.

Outputs:

`RESULT[31:0]`	The result of the multiplication.

a) Design this multiplication unit using one 17×17 signed multiplier and additional logic. (4p)

b) Design this multiplication unit using one 17×17 unsigned multiplier and additional logic. (4p)

c) Which version is best? Why? (You can assume that both kinds of 17×17 multiplier have the same cost in terms of for example area, power usage and latency.) (2p)

Question 5 - AGU (15p)

Draw a schematic and a partial control signal table for an address generator unit.

It should contain five registers:

`AR0[15:0]`	Address register 0
`AR1[15:0]`	Address register 1
`BOTTOM[15:0]`	Lower address for modulo addressing
`TOP[15:0]`	Higher address for modulo addressing
`STEP[15:0]`	Configurable step size

It has the following output:

`ADDRESS[15:0]`	This is the address to the data memory

It has the following inputs:

`IMMEDIATE[15:0]`	Constant data carried by the instruction
`RF[15:0]`	Data from the general register file
`OPERATION[3:0]`	Controls which operation is performed by the AGU according to the table below

`OPERATION`	**Operation to be performed**
`0000`	Load AR0 from general register file
`0001`	Load AR1 from general register file
`0010`	Load STEP from the general register file
`0011`	NOP
`0100`	Direct addressing (the address is based on immediate data in the instruction)
`0101`	Indirect addressing (the address is based on a general purpose register)
`0110`	Post increment of AR0 (step size is based on the STEP register)
`0111`	Post increment of AR1 (step size is based on the STEP register)
`1000`	AR0 + immediate
`1001`	AR1 + immediate
`1010`	Modulo addressing on AR0 with post increment (step size is 1)
`1011`	Modulo addressing on AR1 with post increment (step size is 1)
`1100`	NOP
`1101`	NOP
`1110`	Load bottom register from RF
`1111`	Load top register from RF

Your hardware should support all of these operations, but your control signal table only needs to show the control signal configuration for operation `0000`, `0100`, `0110`, `1000`, and `1011`.

Question 6 - ASIP (15p)

Based on previous experience, a function called Read1bit() can be fairly expensive for an MP3 decoder. Your task is to evaluate the cost of implementing this function in software and hardware. *Hint: The function below is written to simplify assembler implementation, a good hardware implementation may look slightly different.*

```
// Read 1 bit from a bitstream in memory and increment the bitpointer
function Read1bit()
   Memval = memory[CurrentAddress] // The memory is 16 bit wide

   // Calculate 2 to the power of CurrentBit
   Bitmask = 1 << CurrentBit

   // Check if bit number CurrentBit is set in Memval
   if Memval & Bitmask
      Bit = 1
   else
      Bit = 0
   endif

   // Advance bit position (and memory pointer if necessary)
   CurrentBit = CurrentBit + 1
   if CurrentBit > 15
      CurrentBit = 0
      CurrentAddress = CurrentAddress + 1
   endif

   return Bit // Return value is located in R0
endfunction
```

a) Write pseudo assembler code for Read1bit on a typical DSP or RISC processor (like Senior). (2p)

b) Estimate how many clock cycles the Read1bit function will execute in for the best and worst case. (You will need to make and document reasonable assumptions about your processor.) (3p)

c) The Senior pipeline is pictured on the next page in Figure 1). What modules in the pipeline (besides the instruction decoder) do you need to modify to implement Read1bit as one instruction? (It should be possible to execute an arbitrary number of consecutive read1bit instructions without having to stall the pipeline.) (2p)

d) For each module you listed in question c above, describe the modifications you would have to make and include a hardware schematic of the modified parts. (8p)

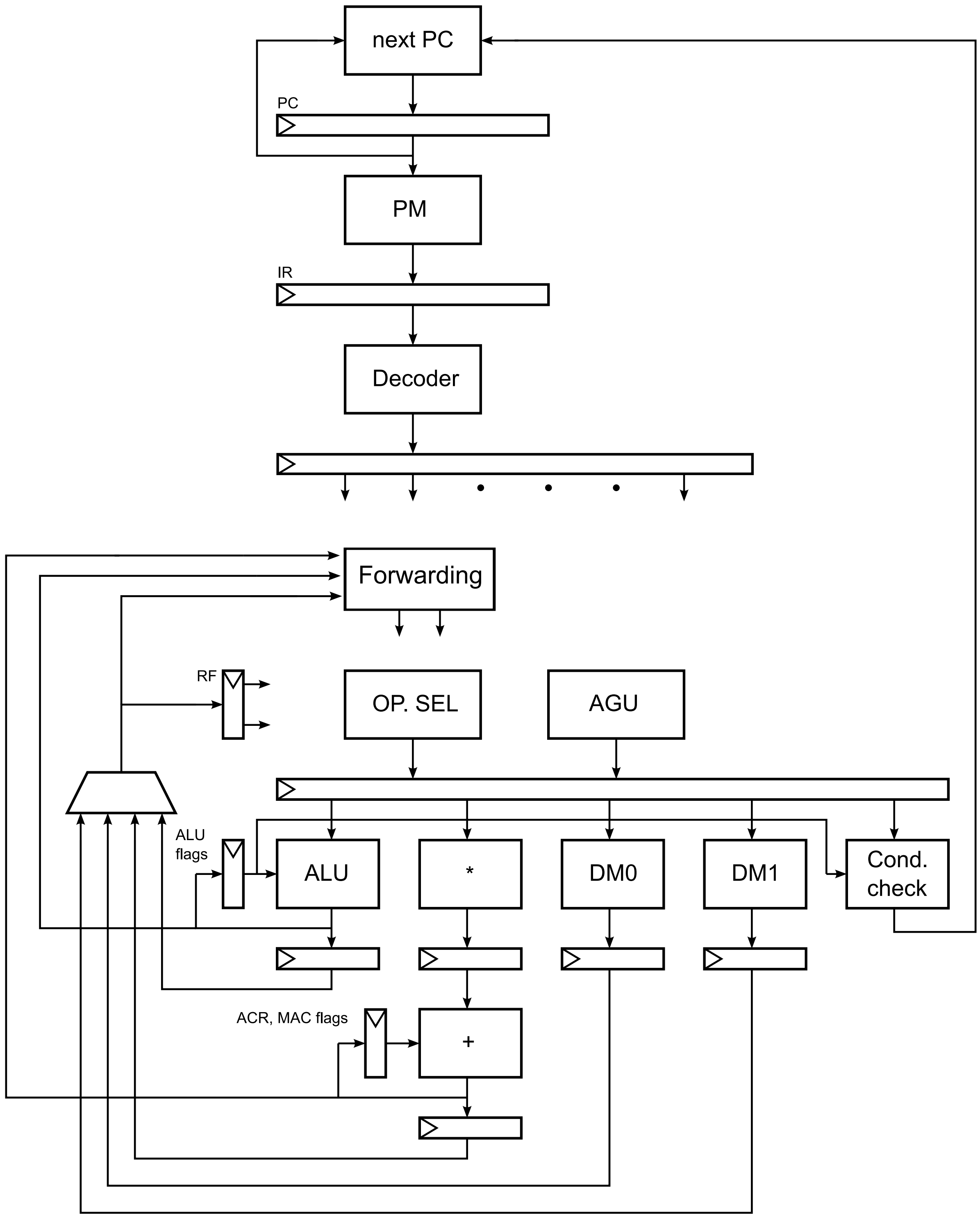

Figure 1: The Senior pipeline