Instructions: ANSWER ALL QUESTIONS in the answer booklets provided. If you make any assumptions, clearly state them in your answer booklet.

Materials Allowed: Course textbook, course lecture notes, ENCS approved electronic calculators.

Location: H539-1
Time: 19:00-22:00

Question 1: [20 points]

The multiply-and-accumulate operation finds use in many digital signal applications. Digital signal processors often include a dedicated instruction as part of their instructions sets which perform the operation. Consider the following three address instruction:

```
MAC  source1, source2, source3/destination
```

This instruction will perform the operation:

```
source3/destination <= (source1 x source2) + source3/destination
```

In other words, the MAC (the mnemonic for *"multiply-and-accumulate"*) instruction multiplies the two specified source operands and adds to result to the third source which also acts as the final destination.

(a) Identify all the **addressing modes** used in the following instruction:

```
MAC  #NUM, (R0)+, R1
```

(b) The sequence of control steps to perform an instruction **fetch** using the single-bus internal CPU organization shown in Figure 1 are:

T0: BUS <= PC, MAR <= BUS, sel_temp = 1, F = ADD, ld_result = 1.

T1: READ = 1, MDR <= data_out from memory (this implies ld_mdr = 1 and sel_mdr = 1),
BUS <= RESULT, PC <= BUS.

T2: BUS <= MDR, IR <= BUS.

In the above, we assume that main memory will respond (for a READ or a WRITE) within one clock cycle.

(i) What advantage does the multiplexer at the input of the MDR offer in terms of program execution speed?

(ii) Rewrite the sequence of control steps needed to perform an instruction **fetch** given that the MDR multiplexer is NOT present (i.e. the MDR is loaded directly from the bus). What modifications would be required to the existing bus organization to accommodate this change?

(c) Give the sequence of control steps to **execute** the `MAC  #NUM, (R0)+, R1` instruction. Assume the existence of the MDR multiplexer when answering this question. You may assume that the instruction occupies one main memory location and that the ALU is capable of performing a multiply operation.

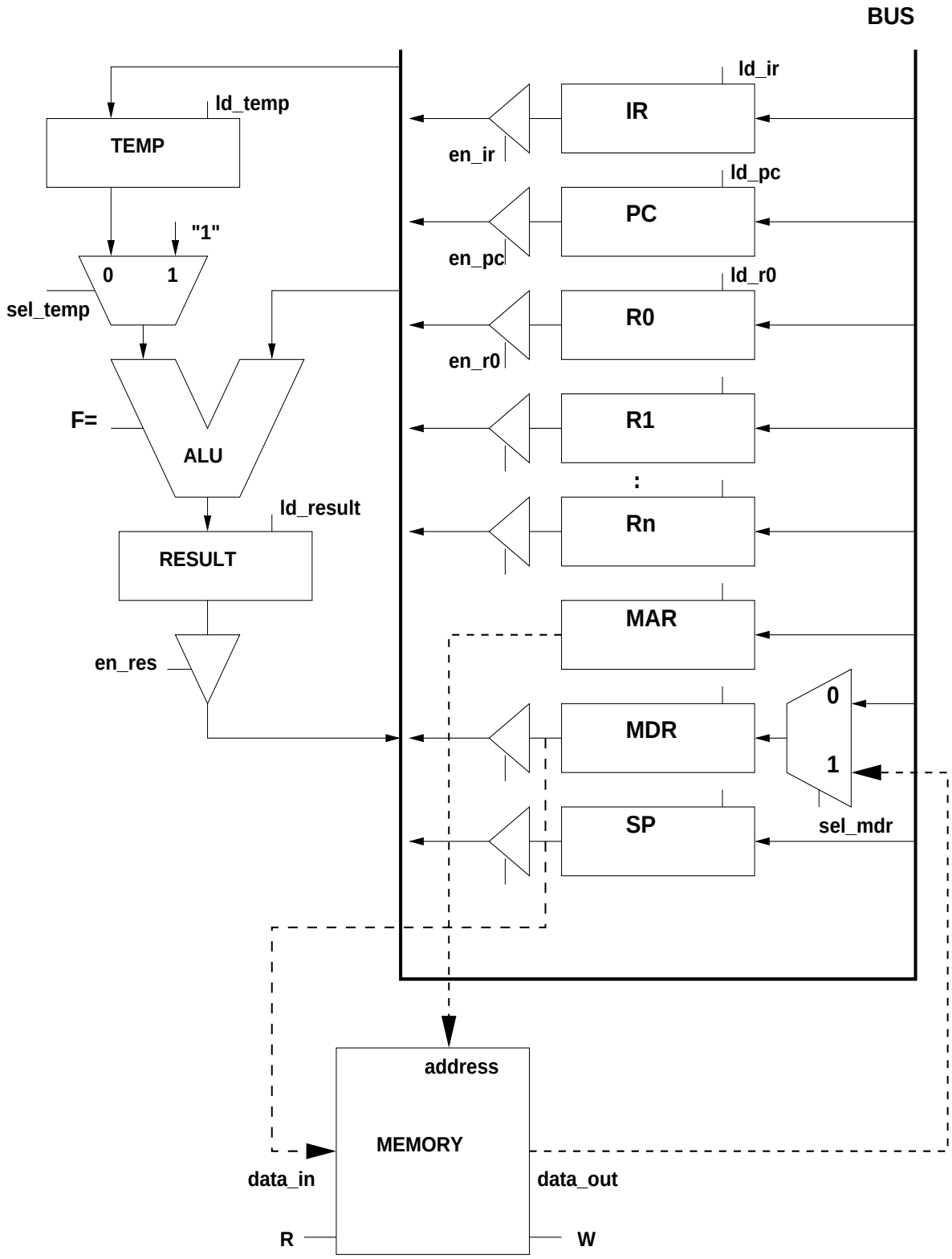

Figure 1: A single-bus internal CPU organization.

Question 2: [20 points]

Consider the following Intel 80x86 assembly language program:

```
section .data

number db  5
answer db  1
section .bss

; put UNINITIALIZED data here using

section .text
        global _start

_start:
       mov esi, number
keith: mov eax, 0
       mov al, [esi]
       mov dl, al
       mov bl, 2
loopy: div bl   ; ax / bl with quotient in al and remainder in ah
       and ax, 1111111100000000b
       cmp ax, 0
       je  there
       inc bl
       cmp bl, dl
       je done
       mov eax, 0      ; restore the  number back into
       mov al, [esi]   ; ax
       jmp loopy
there: mov byte[answer], 0
done:
       mov eax,1       ; The system call for exit (sys_exit)
       mov ebx,0       ; Exit with return code of 0 (no error)
       int 80h
```

(a) State in a few short, clear and concise English sentences the action performed by this program.

An example of a correct method of expressing an answer would be:

"This program removes all vowels from the null-terminated ASCII string stored commencing at memory location number. Vowels are replaced with the blank character. Furthermore, any ASCII characters corresponding to the digits '0', '1', '2'. ... '9' are replaced with their 9's complement value and will appear in inverse video when displayed on a VGA monitor."

Of course, this is meant merely as an example. The program performs something else.

(b) What values will be stored in memory location `answer` if the program is run with value `5` and then value `6` stored in memory location `number`?

Question 3: [20 points]

(a) Consider a small 8 x 3-bit random access memory which makes use of the binary storage cell illustrated in Figure 2. The basic storage cell employs an AND gate at it's output as opposed to using a tri-state buffer.

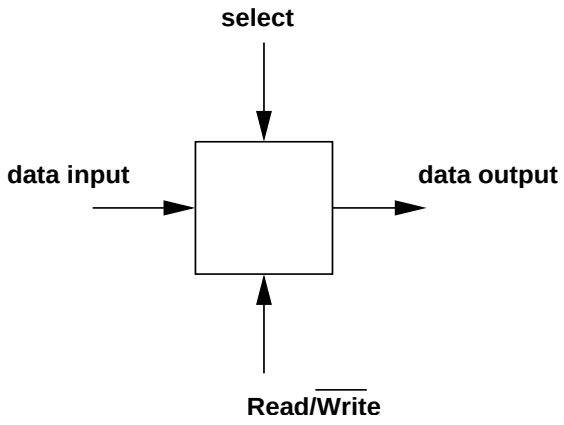

Figure 2: Block diagram of a binary storage cell.

(i) How many binary storage cells would be needed to construct the random access memory?

(ii) Show the internal construction of the random access memory assuming a **1-dimensional** address decoding technique is employed. When drawing your diagram, make use of the block diagram of the binary storage cell and **any other hardware** components required. Use $\mathbf{A_2A_1A_0}$ for the address lines (line $\mathbf{A_2}$ is the most significant address bit) , $\mathbf{d_2d_1d_0}$ for the input data lines ($\mathbf{d_2}$ is the most significant data bit), $\mathbf{dout_2dout_1dout_0}$ for the output data lines, $\mathbf{R/\overline{W}}$ for the read/$\overline{\text{write}}$ line and **enable** for the enable of the memory. Your drawing MUST be clear and legible and indicate all signal names.

(iii) Repeat part(ii) using a **2-dimensional** address decoding technique. Use address lines $\mathbf{A_2A_1}$ to perform **row decoding** and address line $\mathbf{A_0}$ to perform **column** decoding. In the interest of diagram clarity, it is not necessary to show the connections used for the data input signals, nor the R/

W signal. It is also sufficient to merely show the hardware and the signal interconnections necessary to generate the **$dout_0$** output data line. As in part (ii) your drawing MUST be clear and legible.

(iv) Explain in words (or with the aid of a small diagram) what would change in part (ii) and part (iii) if binary storage cells with **tri-state output** capability were employed in the construction of the RAM.

(b) Which of the following give the correct number of SRAM chips of size 1 048 576 locations with 8 bits per location, and the size of the address decoder needed to build a SRAM memory system with a total capacity of 8 589 934 000 locations with 32 bits per location using a 1D organization?

Hint: 2^{20} = 1 048 576 = 1 Mega
$(2^{10})(2^{20})$ = 2^{30} = 1 073 741 000 = 1 Giga
$(2^{3})(2^{10})(2^{20})$ = 2^{33} = 8 589 934 000 = 8 Giga

(i) 8 192 chips with a 32 -to- 8 589 934 000 address decoder.
(ii) 32 768 chips with a 32 -to- 8 589 934 000 address decoder.
(iii) 32 768 chips with a 13 -to- 8192 address decoder.
(iv) Not even Keith has enough money to buy the necessary amount of SRAM chips in order to construct this memory system.

Question 4: [20 points]

(a) The use of a FRAME POINTER register within a CPU allows for a subroutine to:

(i) point to the next element to be popped from the stack.

(ii) point to the bottom of the stack

(iii) access any passed parameters pushed onto the stack by the calling routine and to access any local variables in the stack.

(iv) obtain the address of the next instruction to be executed.

(v) point to itself during a recursive call.

Indicate your answer by choosing one of the above 5 choices.

(b) Express the decimal number `3.1415926` (*pi* to 7 decimal place accuracy) in IEEE-754 32 bit format.

(c) Given the following word (one word = 2 bytes) sized data `0xABCD` (expressed in hexadecimal) show how the data would be stored in a byte-sized main memory for a CPU which uses **high byte** followed by **low byte** versus a machine which stores words in **low byte** followed by **high byte order**.

(d) ROM is volatile. TRUE or FALSE?

(e) A ROM can be re-programmed after it has been manufactured. TRUE or FALSE?

Question 5: [20 points]

Consider a small block-set-associative cache memory with a total of 4 different sets. Each set consists of two blocks. Each block contains 4 locations. Main memory consists of a total of 512 different locations. Answer the following questions for this cache-main memory organization.

(a) Give the format of a main memory address in the form of :

T T T T S S S W W W

Be sure to specify the number of bits required in each field of the main memory address.

(b) How many different main memory blocks maps onto each cache set?

(c) To which cache set does main memory address $(12F)_{16}$ map onto ?

(d) Assume that the tags of the main memory blocks currently residing in cache set 2 are:

`00000` and `10010`.

Indicate whether a cache hit or a cache miss occurs if the CPU generates the following 4 main memory addresses (all addresses are specified in hexadecimal notation) :

```
008
00B
12A
1F8
```

(e) If the block in cache set 2 with tag = `10010` is chosen as the victim block in the case of a cache miss, in which main memory block does this cache block reside in?