le cnam

October 2023

# CPU Architecture

## Exercise set 1

## Objectives

- ✯ Understanding of Addition of unsigned and signed numbers
- ✯ Using registers
- ✯ Understanding Memory

# 1 Addition of Unsigned Numbers

### Exercise 1 Addition of Unsigned Numbers

Let A, B and C be 3 **unsigned** numbers.

1. Compute the following operations using the binary representation
2. Convert A B and C to the decimal representation
3. Compute the following operations using the decimal representation
4. Compare the two results. Did overflows occurred ?

| ① | | Binary | | Decimal |
|---|---|---|---|---|
| A: | | 0110 0111 | | |
| B: | + | 0000 1001 | + | |
| C: | = | | = | |

| ② | | Binary | | Decimal |
|---|---|---|---|---|
| A: | | 0010 1011 | | |
| B: | + | 0011 1001 | + | |
| C: | = | | = | |

| ③ | | Binary | | Decimal |
|---|---|---|---|---|
| A: | | 0110 1011 | | |
| B: | + | 0011 1001 | + | |
| C: | = | | = | |

| ④ | | Binary | | Decimal |
|---|---|---|---|---|
| A: | | 1111 1111 | | |
| B: | + | 0000 0001 | + | |
| C: | = | | = | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| ⑤ | | Binary | Decimal | ⑥ | | Binary | Decimal |
| A: | | 1110 1011 | | A: | | 0110 0110 | |
| B: | + | 0001 1001 | + | B: | + | 1001 1001 | + |
| C: | = | | = | C: | = | | = |

## 2 Addition of Signed Numbers

### Exercise 2 Addition of Signed Numbers

Let A, B and C be 3 **signed** numbers.

1. Compute the following operations using the binary representation
2. Convert A B and C to the decimal representation
3. Compute the following operations using the decimal representation
4. Compare the two result. Did overflows occurred ?
5. What is difference do you see compared to exercise 1?

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| ① | | Binary | Decimal | ② | | Binary | Decimal |
| A: | | 0110 0111 | | A: | | 0010 1011 | |
| B: | + | 0000 1001 | + | B: | + | 0011 1001 | + |
| C: | = | | = | C: | = | | = |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| ③ | | Binary | Decimal | ④ | | Binary | Decimal |
| A: | | 0110 1011 | | A: | | 1111 1111 | |
| B: | + | 0011 1001 | + | B: | + | 0000 0001 | + |
| C: | = | | = | C: | = | | = |

| ⑤ | Binary | Decimal | ⑥ | Binary | Decimal |
|---|---|---|---|---|---|
| A: | 1110 1011 | | A: | 0110 0110 | |
| B: | + 0001 1001 | + | B: | + 1001 1001 | + |
| C: | = | = | C: | = | = |

## Exercise 3 Simple Adder

1. Give the truth table of the addition of 2 bits: $s = a + b$
2. Give the Boolean expression of $s$
3. Give the truth table of the carry $c$ resulting of the addition
4. Give the Boolean expression of $c$
5. Draw the schematics of $s$ and $c$ using logic gates (see previous exercise sheet)

## Exercise 4 Full Adder

1. Give the truth table of the addition of 2 bits and the carry of preceding rank: $s = a + b + c_{in}$
2. Give the Boolean expression of $s$
3. Give the truth table of the carry $c_{out}$ resulting of the addition
4. Give the Boolean expression of $c_{out}$
5. Draw the schematics of $s$ and $c_{out}$ using logic gates

## Exercise 5 Increment

The circuit on figure 1 represent an *increment* operator: each cycle, the register value is increased by `Cin`. The register is a 8 bits registers (1 byte).
Fill up the following digital timing diagram:

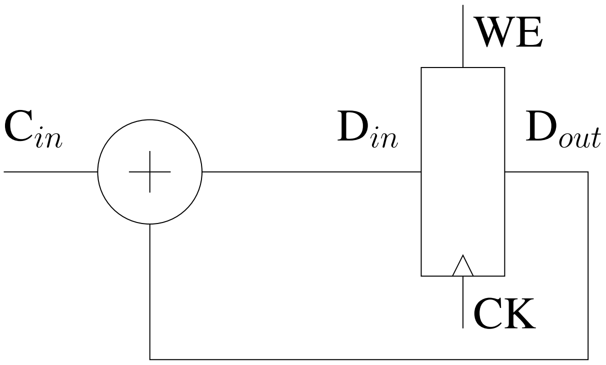
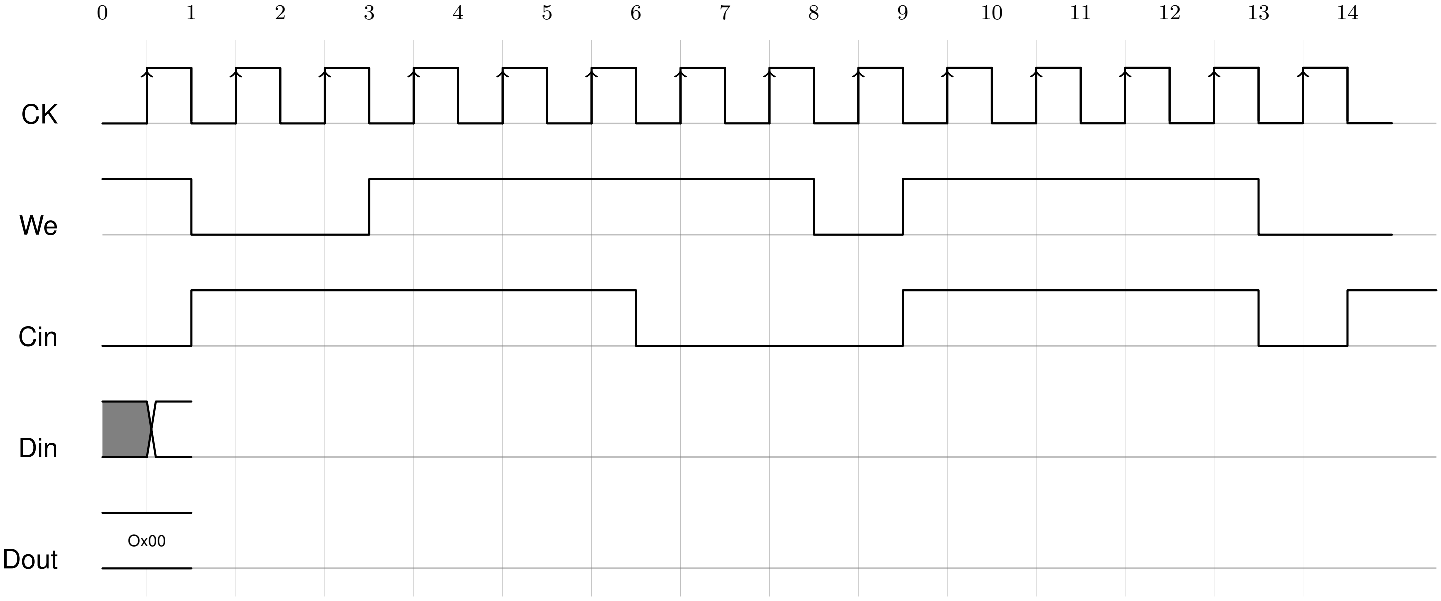


Figure 1: Increment operator



1. At which cycle `Din` is 4 ?
2. At which cycle `Dout` is 6 ?

## Exercise 6 Moving data to the ALU

We consider the circuit represented on figure 2. In that circuit, registers $RA$, $RB$, $RC$ and $RD$ are connected to a Arithmetic and Logical Unit (ALU). The ALU is able to perform the addition and subtraction between its inputs and at each operation it sets the following flags:

- $CF$ : carryflow or the carry out bit
- $OV$ : overflow or capacity exceeded on relative numbers
- $SF$ : signflow or bit sign of the result
- $ZF$ : zeroflow or signal reporting a zero result

We consider that all registers and data path inside the ALU are 4 bits wide. Registers can be loaded with external values or ALU result.

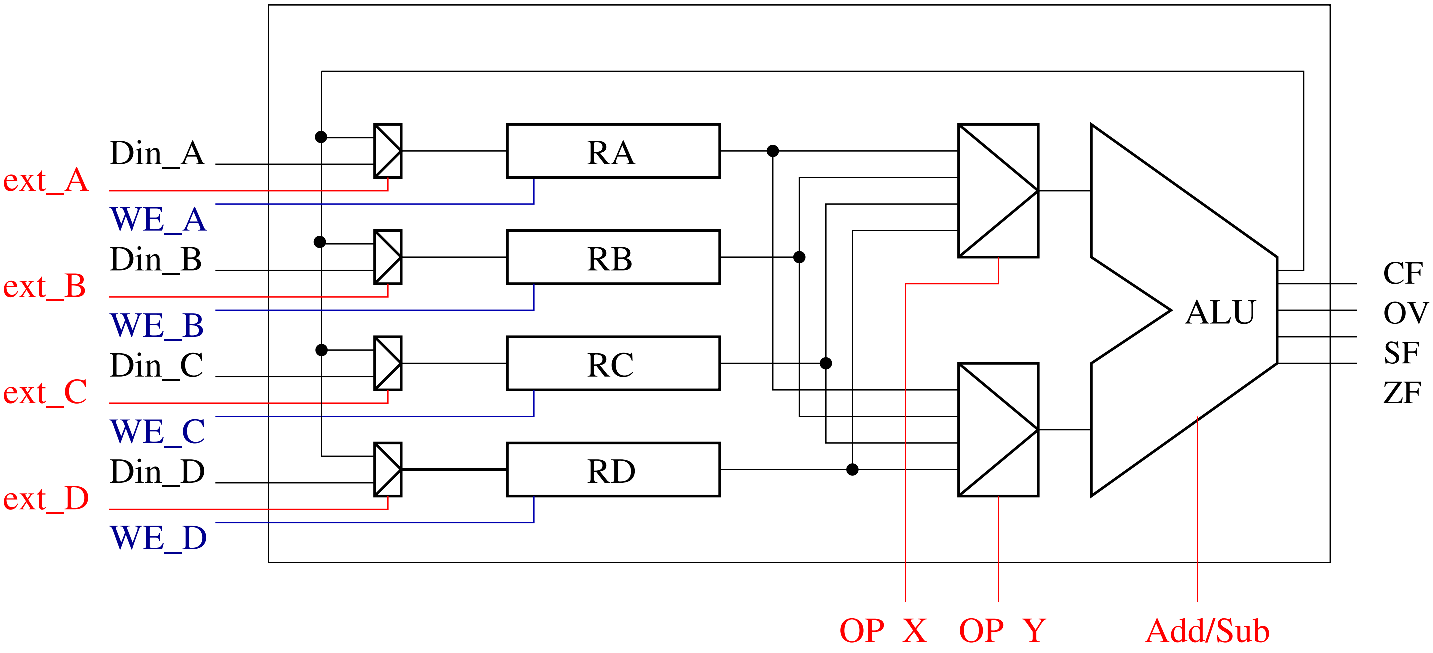


Figure 2: Data Path

Signal commands for the different units are:

- For each register $X$ :
  - the signal `ext_X` allow to select the input value `Din_X` or the output of the ALU: if `ext_X = 1` then `Din_X` is selected. Else the ALU output is selected
  - the Write Enable signal `WE_X` is the signal which allow writing in the register $X$: if `WE_X = 1` then writing is allowed in register $X$.
- For ALU :
  - command `OP_X` and `OP_Y` allow to select input registers of the ALU. Commands `OP_X` and `OP_Y` allow to select a register within $RA$, $RB$, $RC$ and $RD$ using a value encoded on 2 bits: 00 → $RA$, 01 → $RB$, 10 → $RC$, 11 → $RD$
  - ALU Operation is selected by `Add/Sub` command. When $Add/Sub = 1$ addition operation is selected. When $Add/Sub = 0$ subtraction operation is selected (`OP_X - OP_Y`)

Data are on 4 bits as are register, multiplexer and ALU inputs and outputs. The clock does not appear on the figure 2 but is a implicit input of the registers.

**Question 1**

On figure 3, specify which wires correspond to data on 1 bit, 2 bits and 4 bits.

The ALU needs a little bit less than 1 cycle to compute the operation. So, when setup the ALU inputs around (a little bit before and a little bit after) a raising edge of the clock, we need to wait the raising edge of the next cycle $i + 1$ before the result could be written to a register.

A new operation could start at the raising edge of cycle $i + 1$ while the result of the operation at cycle $i$ is written to a register.

Moreover, flags $CF$, $OV$, $SF$ and $ZF$ are stored in the *Status Register* (not shown on figure 2). That register could be used to choose the next action to be performed. As for the registers, flag values are stored at cycle $i + 1$ (on raising edge) for the operation done at cycle $i$.

### Question 2

Fill up the digital timing diagram on figure 3 for command signal when the following sequence of instructions is given:

1. Put the value $0x7$ in $RA$ (indication: first put the value $0x7$ on `Din_A` then enable writing on register $RA$)
2. Put the value $0xB$ in $RB$
3. Put the result of the operation $RA + RB$ in $RC$ (indication: first put $RA$ and $RB$ value on ALU inputs `OP_X` and `OP_Y` while addition operation, then write the ALU result in $RC$ on next cycle)
4. Put the result of $RA - RB$ in $RD$

Does some of this operation could be done in the same cycle ?

## Exercise 7 Memory

We consider a memory of 8 words of 4 bytes. The addressable unit is the byte. Transfers can be done on words (32bits), half-words (16bits) or bytes (8bits). The bus between the processors and the memory has 3 sets of wires: address, command and data. The data wire sets is shared between reading and writing operations. This configuration is shown on figure 4.

### Question 1

What is the purpose of this 3 sets of wires: address, command and data ? Deduce the width of wire sets address and data, then for the command wire set. Give an encoding for each command.

We suppose the memory organisation is little-endian: when a multi bytes transfer occurs, the least significant bytes is stored at the smallest address. The address are aligned: for a transfer of $n$ bytes, address are aligned on multiple of $n$ bytes (for $n = 2$ or $n = 4$).

### Question 2

Specify all the values on wire sets and the memory location of data when the following transfers occur:

1. write the word $0xFEDCBA98$ starting at address $0x00$
2. write the word $0x76543210$ starting at address $0x04$
3. write the half-word $0x3210$ starting at address $0x02$

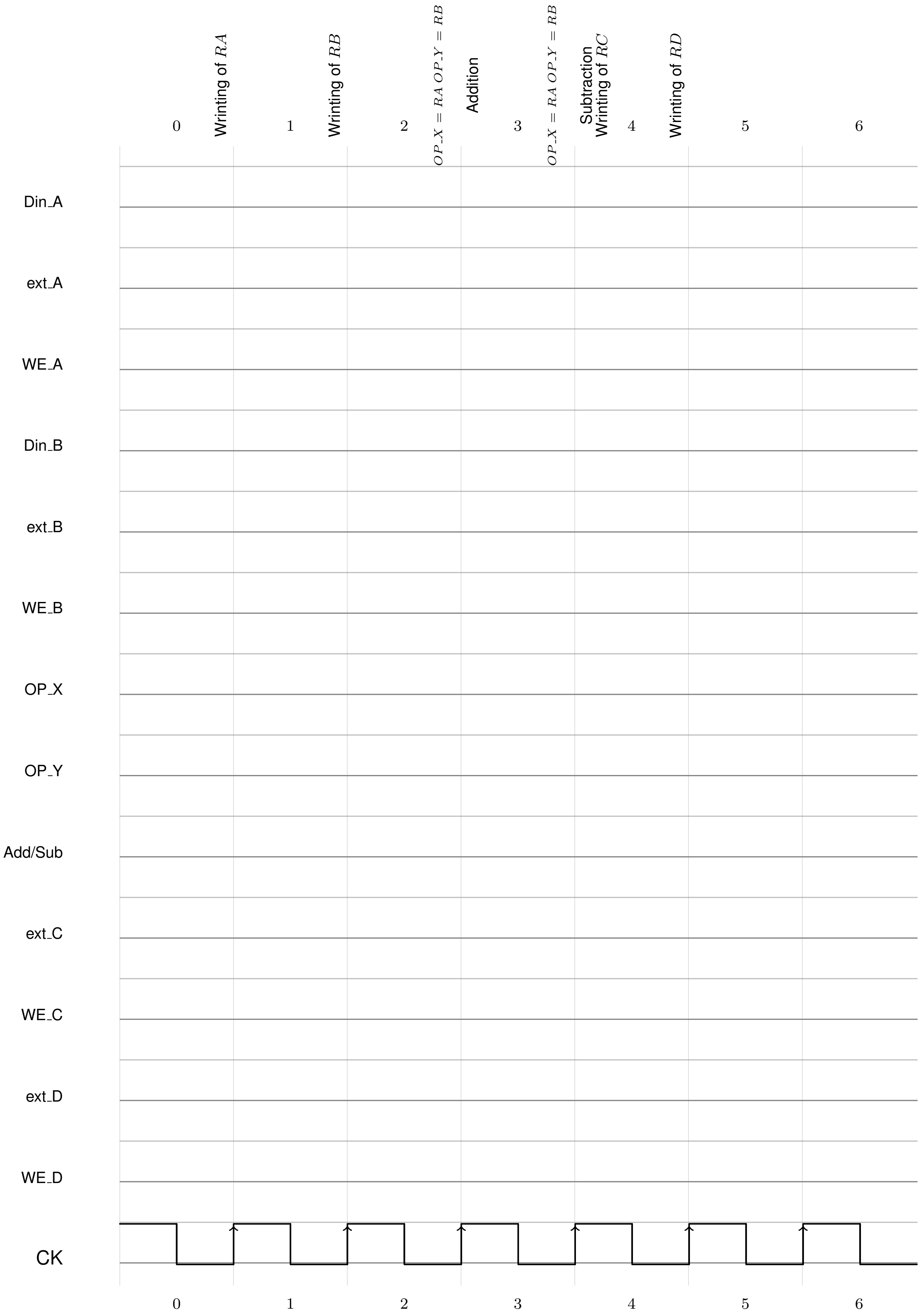


Figure 3: Data path digital timing diagram

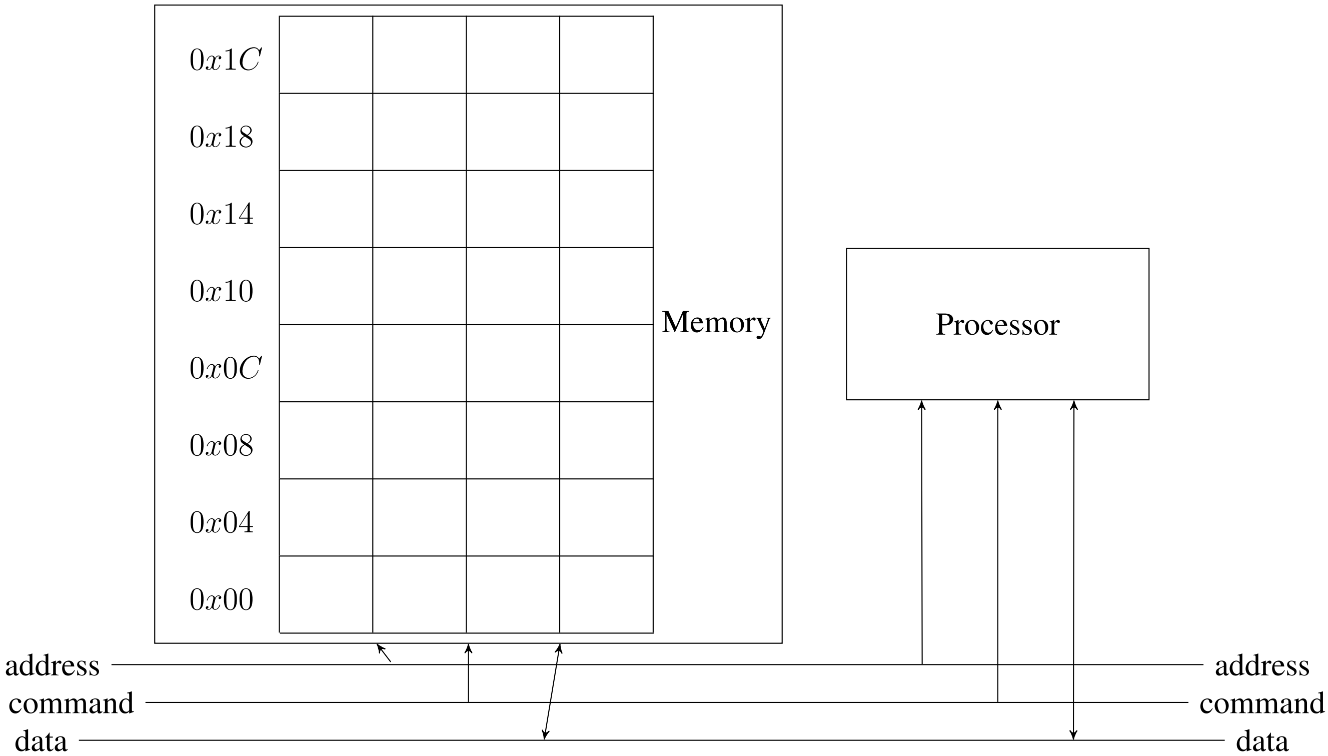


Figure 4: Memory of 8 words of 4 bytes connected to the processor