QUIZ-2 CMPE-523 05.01.2018 (3 points, 120 min)

St. Name, Surname_____ St.Id#_____

Instructor Alexander Chefranov

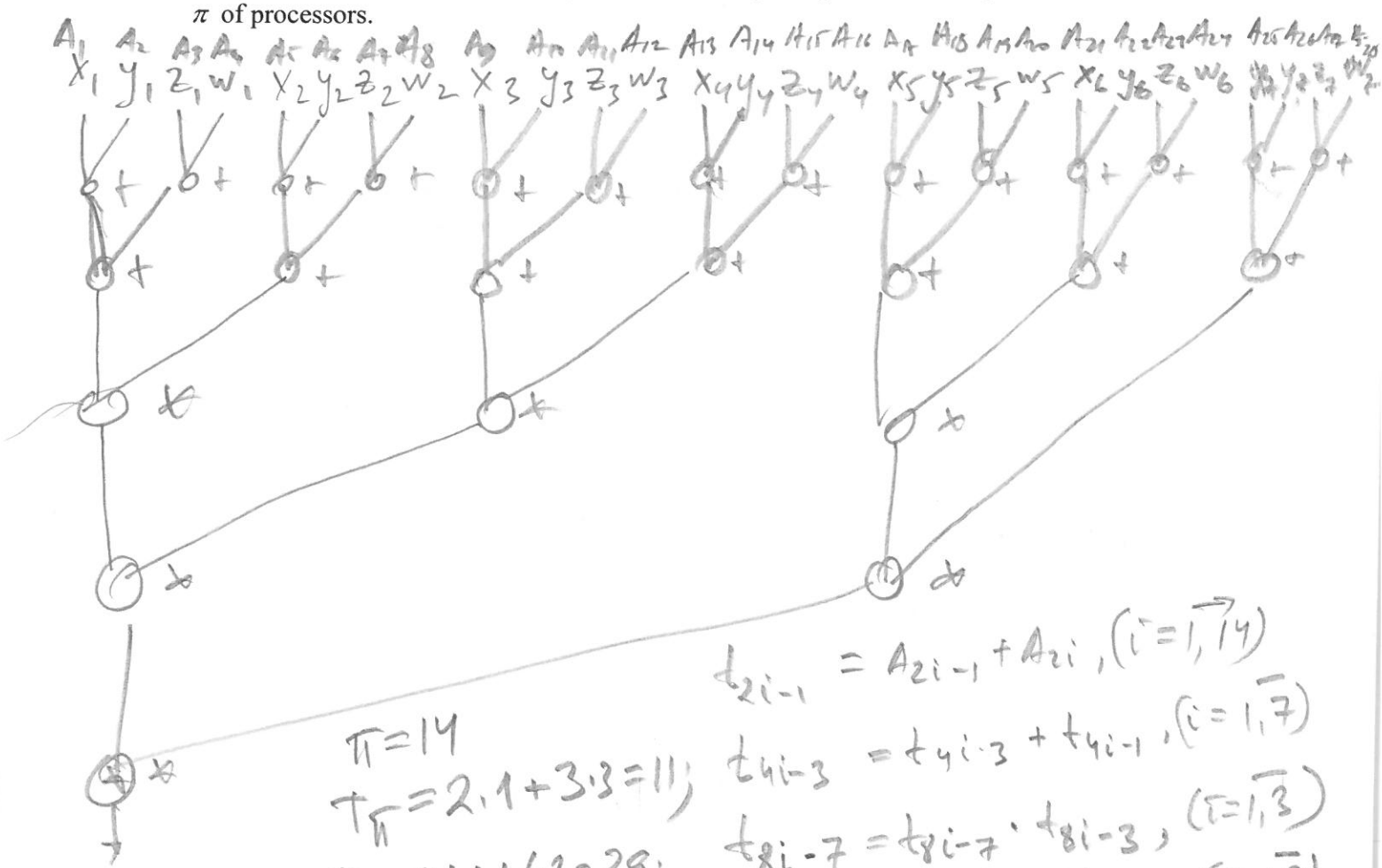
Totally 7 tasks, 3 points, 8 pages

Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7	Total
0.5	0.5	0.4	0.4	0.4	0.4	0.4	3

Task 1. (0.5 points). Using associativity, draw the flattest possible dependence graph for the following product calculation

$$\prod_{i=1}^{7}(z_i + x_i + y_i + w_i).$$

Write SIMD pseudocode for its calculation. Assume that addition takes 1 time unit, and multiplication takes 3 time units. What is the minimal number π of processors providing maximal performance for that program? Estimate speedup and efficiency for that number π of processors.



$t_{2i-1} = A_{2i-1} + A_{2i}, (i = \overline{1,14})$

$t_{4i-3} = t_{4i-3} + t_{4i-1}, (i = \overline{1,7})$

$t_{8i-7} = t_{8i-7} \cdot t_{8i-3}, (i = \overline{1,3})$

$t_{16i-15} = t_{16i-15} \cdot t_{16i-7}, (i = \overline{1,2})$

$t_1 = t_1 \cdot t_{17}$

$\pi = 14$

$T_\pi = 2 \cdot 1 + 3 \cdot 3 = 11;$

$T_1 = 21 \cdot 1 + 6 \cdot 3 = 39;$

$S_\pi = \frac{39}{11}; \quad E_\pi = \frac{31}{14 \cdot 14};$

$A_{4i-3} = y_i, A_{4i-2} = y_i, A_{4i-1} = z_i, A_{4i} = w_i, i = \overline{1,7}$

Task 2. (0.5 points). Consider the code below

```
For i:=1 step 1 until N
  For j:=1 step 1 until N
    C[I,j]:=0;
For k:=1 step 1 until N
  For i:=1 step 1 until N
    For j:=1 step 1 until N
      C[I,j]:=c[I,j]+A[I,k]*B[k,j];
```

What problem is solved by the code?
Assuming a SIMD computer has N^2 processing elements, write respective parallel pseudo-code.

Matrix product $C = AB$ is calculated

$c_{ij} = 0$, $(1 \le i,j \le N)$

for $k = 1$ step 1 until N

$c_{ij} = c_{ij} + a_{ik} \cdot b_{kj}$, $(1 \le i,j \le N)$

Task 3 (0.4 points) Write SIMD assembly code to calculate absolute value of the result of the sum of two vectors (SIMD assembly language instructions are on the last page). Assume the number of the each vector components is 15, and the number of processing elements is 6. Specify memory layout (distribution of vectors over memory blocks). Give necessary explanations.

```
E       data -1
Zero    data 0
A       BSS  3×6
B       BSS  3×6
ABS     BSS  3×6
ZR      BSS  1×6
M       equiv 1
PE      equiv 2
I       equiv 3
        ldxi  I, 0
        ldxi  M, 15
        ldxi  PE, 0
        cload Zero
        cbcast
        movR  A
        sta   ZR
Loop:   lod   A, I
        add   B, I
        stmask
        movemask to 4
        cet   ZR
        mask
        cload E
        cbcast
        rmiR
        move4 tomask
        mask
        <1    ABS
```

X(B) = 0 row A
X(1) = 15
X(2) = 0

AC = 0
$R_k = 0$
$A_k = R_k$
$ZR_k = 0$
$A_k = A$
$A_k = A + B$

X(4) = mask
A+B < 0

AC = -1
$R_k = -1$
(A+B) · (-1), where A+B<0
restore mask

ABS = |A+B|

incx I,1 I = I+1
vecloop PE, M, Loop

Task 4 (0.4 points) Consider the code below

Assembly Language

```
N       DATA    64          Number of PEs and matrix size.
ZRO     DATA    0.0         Constant.
I       EQUIV   2           Row number is in index 2,
LIM     EQUIV   1             and the limit is in index 1.
B       BSS     64×64       Storage for the matrix.
        ldxi    I,0         Set I to start of columns.
        cload   ZRO         Set all
        cbcast              PE accumulators
        movR    toA           to zero.
        ceq     B           Test first row for zeros.
        not     mask, mask  Enable processors
        mask                  corresponding to nonzeros.
        ldx     LIM,N       Set up loop limit.
ILP     lod     B,I         Fetch row I and
        div     B             scale it, and
        sto     B,I             store it, if scale factor ≠ 0.
        incx    I,1         Increment index and
        cmpx    I,LIM,ILP     loop if not complete.
```

Figure 3-16
Column scaling using the mask register.

Trace the code assuming N=4 instead of 64 (specify particular numbers to work with). Explain your tracing. Decide what the code is doing in the terms of operations on matrices and illustrate your answer by a numerical example. Give necessary explanations.

The code divides the 1st row of B by itself (where ≠0), getting all 1's and 0's. Then, the next rows are divided by 1's not changing the dividends.

X(2)	A_1	A_2	A_3	A_4	R_1	R_2	R_3	R_4	B_{11}	B_{12}	B_{13}	B_{14}	B_{21}	B_{22}	B_{33}	B_{34}
									1	3	3	4	0	1	2	0
ldxi 0																
cload	CA = 0															
cbcast					0	0	0	0								
movRtoA	0	0	0	0												
ccq B	mask = 0100															
not mask mask	mask = 1011															
mask N	Sel = 1011															
ldx LIM,N	X(1) = 4															
lod B,I	1		3	4												
div B	1		1	1					1	0	1	1				
sto B,I																
incx I,1	1															
cmpx I, LIM, ILP	1 < 4 → PC = address of ILP (goto ILP)															
lod B,I	0		2	0												
div B	0		2	0					0	1	2	0				
sto B,I																
incx I,1	2															
cmpx I, LIM, ILP	2 < 4 → goto ILP															

Example: B = $\begin{matrix} 1 & 0 & 3 & 4 \\ 0 & 1 & 2 & 0 \\ 2 & 1 & 1 & 5 \end{matrix}$ → $\begin{matrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 2 & 0 \\ 2 & 1 & 1 & 5 \end{matrix}$

Task 5 (0.4 points) Consider FORTRAN-90 array A(1:20,1:12,1:30). Let B=A(3:10:3,1,:). For B, define its dimension, lower boundary, upper boundary, extent in each dimension, and the total number of elements. Give necessary explanations.

B is the 1st dimension : low = 3 up = 10 ext = 8
2nd dim : 1, 1, 1 → no dimension
3rd dim : at in A beginning of : →
low = 1 up = 30, ext = 30
Total # = 8 · 1 · 30 = 240 elements

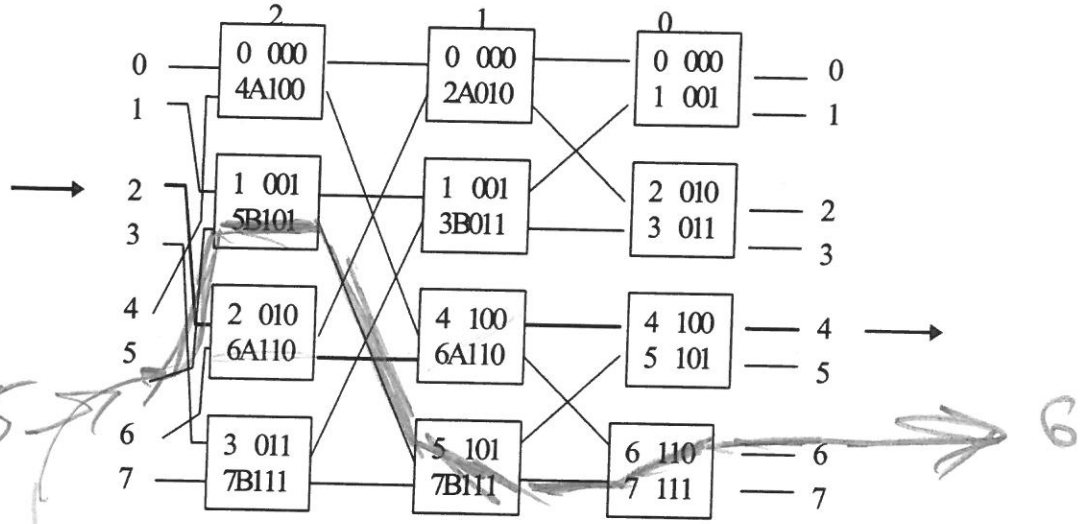
Task 6 (0.4 points) Assume, N=10 similar processes, P1,.., P10, share a printer. Write a semaphore solution for the i-th process, Pi, allowing sharing without the printer without mixing documents coming from different processes. Give necessary explanations.

```
Sem S = 1;  // binary semaphore to 1

Pi :
    :
    P(S);   acquire semaphore
    print;
    V(S);   release semaphore
    :
```

Task 7 (0.4 points)

Consider the following hypercube interconnection network



What routine tag shall be used for connecting input 5 with output 6. Give necessary explanations.

$S = 5 = 101$

$D = 6 = 110$

Routine tag $= S \oplus D = 101 \oplus 110 = 011$

When routing, the 1st switch works in straight mode, the 2nd exchanges, and the 3rd one exchanges. The route is shown

Instruction	Assembly code		Action
Vector load	lod	a, index, i	$S_k \rightarrow A_k \leftarrow M_k[pea_k], (0 \le k \le N-1)$
Vector store	sto	a, index, i	$S_k \rightarrow M_k[pea_k] \leftarrow A_k, (0 \le k \le N-1)$
Vector add	add	a, index, i	$S_k \rightarrow A_k \leftarrow A_k + M_k[pea_k], (0 \le k \le N-1)$
Vector subtract	sub	a, index, i	$S_k \rightarrow A_k \leftarrow A_k - M_k[pea_k], (0 \le k \le N-1)$
Vector multiply	mul	a, index, i	$S_k \rightarrow A_k \leftarrow A_k \times M_k[pea_k], (0 \le k \le N-1)$
Vector divide	div	a, index, i	$S_k \rightarrow A_k \leftarrow A_k / M_k[pea_k], (0 \le k \le N-1)$
Broadcast	bcast	index	$S_k \rightarrow R_k \leftarrow R_{X[index]}, (0 \le k \le N-1)$
Move PE register	mov $\begin{Bmatrix} A \\ R \\ I \end{Bmatrix}$	to $\begin{Bmatrix} A \\ R \\ I \end{Bmatrix}$	$S_k \rightarrow \begin{Bmatrix} A_k \\ R_k \\ I_k \end{Bmatrix} \leftarrow \begin{Bmatrix} A_k \\ R_k \\ I_k \end{Bmatrix}, (0 \le k \le N-1)$
Register add	radd		$S_k \rightarrow A_k \leftarrow A_k + R_k, (0 \le k \le N-1)$
Register subtract	rsub		$S_k \rightarrow A_k \leftarrow A_k - R_k, (0 \le k \le N-1)$
Register multiply	rmul		$S_k \rightarrow A_k \leftarrow A_k \times R_k, (0 \le k \le N-1)$
Register divide	rdiv		$S_k \rightarrow A_k \leftarrow A_k / R_k, (0 \le k \le N-1)$

Figure 3-6
Set of vector instructions for an SIMD machine.

Instruction	Assembly code		Action
Load index	ldx	ix2, a.index	$X[ix2] \leftarrow M[ca]$;
Store index	stx	ix2, a.index	$M[ca] \leftarrow X[ix2]$;
Load index immediate	ldxi	ix2, a.index	$X[ix2] \leftarrow ca$;
Increment index	incx	ix2, a.index	$X[ix2] \leftarrow X[ix2] + ca$;
Decrement index	decx	ix2, a.index	$X[ix2] \leftarrow X[ix2] - ca$;
Multiply index	mulx	ix2, a.index	$X[ix2] \leftarrow X[ix2] \times ca$;
Load data	cload	a, index	$A \leftarrow M[ca]$;
Store data	cstore	a, index	$M[ca] \leftarrow AC$;
Compare and branch	cmpx	index.ix2, a	$(X[index] \le X[ix2]) \rightarrow PC \leftarrow ca$;

Figure 3-7
SIMD control unit instruction set.

Instruction	Assembly code		Action
Compare <	clt	a, index, i	$S_k \rightarrow (M_k[pea_k] < A_k) \rightarrow mask(k) \leftarrow 1, (0 \le k \le N-1)$;
Compare =	ceq	a, index, i	$S_k \rightarrow (M_k[pea_k] = A_k) \rightarrow mask(k) \leftarrow 1, (0 \le k \le N-1)$;
Compare >	cgt	a, index, i	$S_k \rightarrow (M_k[pea_k] > A_k) \rightarrow mask(k) \leftarrow 1, (0 \le k \le N-1)$;
...	...		...
Mask PEs	mask		$S_k \leftarrow mask(k), (0 \le k \le N-1)$;
Save enables	stmask		$mask(k) \leftarrow S_k, (0 \le k \le N-1)$;
CU move	move $\begin{Bmatrix} i \\ m \\ AC \end{Bmatrix}$	to $\begin{Bmatrix} i \\ m \\ AC \end{Bmatrix}$	$\begin{Bmatrix} X[j] \\ mask \\ AC \end{Bmatrix} \leftarrow \begin{Bmatrix} X[j] \\ mask \\ AC \end{Bmatrix}$;

Figure 3-11
Cooperative SIMD instructions involving the mask.

Instruction	Assembly code		Action
Bitwise AND	and	ix1, ix2, ix3	$X[ix1] \leftarrow X[ix2]$ *and* $X[ix3]$;
Bitwise OR	or	ix1, ix2, ix3	$X[ix1] \leftarrow X[ix2]$ *or* $X[ix3]$;
Bitwise NOT	not	ix1, ix2	$X[ix1] \leftarrow$ *not* $X[ix2]$;
...	...	...	...

Figure 3-12
Vector logic instructions useful for complex conditionals.

Instruction	Assembly code	Action
Broadcast *AC*	cbcast	$S_k \rightarrow R_k \leftarrow AC, (0 \le k \le N-1)$;

Figure 3-8
A simple cooperative instruction.

Instruction	Assembly code		Action
Shift routing	route	a, index	$(S_k \rightarrow (R_{(k+ca) \bmod N} \leftarrow R_k), 0 \le k \le N-1)$;

Figure 3-19
A minimal set of routing instructions.