

TEMA 2: LLENGUATGE MÀQUINA

• ESPAI de memòria: ESPAI LINEAL 2^{32} posicions d'1 byte. Mode platenit / LITTLE ENDIAN.

• REGISTRES:

- 32b: %eax, %ebx, %ecx, %edx, %esi, %edi, %esp, %ebp, %eip, %efl, %ss
- 16b: %ax, %bx, %cx, %dx, %si, %di, %sp, %bp
- 8b: %ah, %al, %bh, %bl, %ch, %cl, %dh, %dl

• MODES DIRECCIONAMENT:

- IMMEDIAT: 1, 2 o 4 bytes. \$19, \$-3, \$0x2A, \$0x2A45

- REGISTRE: %eax, %ah, %esi REG. 32b.

- MEMÒRIA: D(Rb, Ri, s) \rightarrow M[Rb + Ri * s + D] 1,2,4 bytes
FACTORS
ESCALA
1,2,4,8

- CODIFICACIÓ INSTRUCCIONS: 1/2 bytes
OPCODE 1 byte
MODE 1 byte
sIB 1/24 bytes
Desp. - Immediat. ->

OPERACIÓ, MIDA OPERA.
OPERANT FONT i DESTÍ:
si OP. Font ò imm o REG/IR.

mode direcció op.
men, el reg. d'op.
registe i si hi ha desplaçament si un op. memòria.

EXEMPLES:

- 3(%eax, %ebx) \rightarrow M[eax+ebx-3]
- (%eax, %ebx, 3) \rightarrow M[eax+ebx*3]
- (%ebx, 4) \rightarrow M[ebx*4]
- 3(%eax, %esi, 2) \rightarrow M[eax+esi*2+3]
- %al \rightarrow 8b memòries %eax

• INSTRUCCIONS:

- MOVIMENT DADES:

- movsx op1, op2 \leftarrow op2 \leftarrow op1
- movsw op1, op2 \leftarrow extsis(op1)
- movz op1, op2 \leftarrow extzero(op1)
- pushl op1 \leftarrow %esp \leftarrow %esp - 4; M[%esp] \leftarrow op1
- popl op1 \leftarrow M[%esp]; %esp \leftarrow %esp + 4
- leal op1, op2 \leftarrow &op1

EXEMPLIC:

- movb \$-1, %al
- movsbw %ch, %ax
- movzw1 %bx, %edx
- push 12(%ebp)
- pop %eax
- leal (%ebx, %ecx), %eax // op1 memòria

- ARITMÈTIQUES:

- addx op1, op2 \leftarrow op2 \leftarrow op1
- subx op1, op2 \leftarrow op2 \leftarrow op1
- adcx op1, op2 \leftarrow op2 \leftarrow op1 + CF
- sbbx op1, op2 \leftarrow op2 \leftarrow op1 - CF
- incx op1 \leftarrow op1 + 1
- decx op1 \leftarrow op1 - 1
- negx op1 \leftarrow !op1
- imul op1, op2 \leftarrow op2 \leftarrow op1
- imul imm, op1, op2 \leftarrow op2 \leftarrow op1 * imm
- imull op1 \leftarrow %edx * %eax = op1 * %eax
- null op1 \leftarrow %edx * %eax = op1 * %eax
- cltd \leftarrow %edx * %eax = exts(%eax)
- idivl op1 \leftarrow %eax = edx * eax / op1
- idivl op1 \leftarrow %eax = edx * eax / op1
- divl op1 \leftarrow %eax = edx * eax / op1

EXEMPLIC:

- addl \$13, %eax // op2 REG.
- subw %cx, %ax // imm cte.
- adcl %edx, %eax // op1 mem/res, integers
- sbbi %dx, %ax // op op
- incl %eax
- decw %bx
- negl %eax
- inul (%ebx) %eax // op2 res
- imul \$3, %eax, %ecx // imm cte.
- imull (%ebx) // op1 mem/res, integer
- null (%eax) // op1 mem/les, natural
- cltd
- idivl (%ebx) // op1 mem/les, integer
- divl %eax // op1 mem/les, natural

- LÒGICQUES:

- andx op1, op2 \leftarrow op2 & op1
- orx op1, op2 \leftarrow op2 | op1
- xorx op1, op2 \leftarrow op2 ^ op1
- notx op1 \leftarrow op1 = ~op1
- salx K, op1 \leftarrow op1 <<= K (Alt)

EXEMPLIC:

- andl \$13, %eax
- orw %cx, %ax
- xorl %edx, %eax
- notb %ah
- salv \$2, %eax // K: imm o %cl

- SHLX K, op1
- SARX K, op1
- SHRZ X K, op1
- COMPX op1, op2
- TESTX op1, op2

$op1 \ll K \text{ (los)}$
 $op1 \gg K \text{ (arit)}$
 $op1 \gg K \text{ (los)}$
 $op2 - op1$
 $op2 \& op1$

SHLW Y.CL, Y.DX // K: imm / Y.CL
 SARL \$1, Y.EAX // K: imm / Y.CL
 SHRW Y.CL, Y.DX // K: imm / Y.CL
 CPL \$13, Y.EAX // FLAGS
 TESTW Y.CX, Y.AX // FLAGS

- Seqüènciament:

- JUP etiq.
- JMP op.
- JCC etiq.
- Jcc etiq.
- Jcc etiq.
- CALL ETIQ.
- CALL OP.
- RET

$EIP += \text{despl.}$
 $EIP = op$
 $\text{if}(cc) EIP += \text{despl.}$
 $\text{if}(cc) EIP += \text{despl.}$
 $\text{if}(cc) EIP += \text{despl.}$
 $Y.ESP -= 4;$
 $M[Y.ESP] = EIP;$
 $EIP += \text{despl.};$
 $Y.ESP -= 4;$
 $M[Y.ESP] = EIP;$
 $EIP = op; ;$
 $EIP = M[Y.ESP];$
 $Y.ESP -= 4$

EXAMPLE:

JMP LOOP // EIP = Retiq.
 JMP(Y.EBX, Y.ESI, 4) // op: reg/mem
 JLE ELSE // Z
 JA LOOP // N
 INC ERROR // FLAGS
 CALL SUB // GUARDAR @retora
 $EIP = \text{Retiq.}$
 CALL (Y.EBX) // op: reg/mem
 RET

- FLAGS:

- Je etiq.
- Jne etiq.
- JS etiq.
- JNS etiq.
- JG etiq.
- JNS etiq.
- JL etiq.
- JNL etiq.
- JA etiq.
- JAE etiq.
- JB etiq.
- JBE etiq.

ZF // IGUAL / ZERO
 $\neg ZF$ // NO IGUAL / NO ZERO
 SF // NEGATIU
 $\neg SF$ // NO NEGATIU
 $\neg(SF \wedge OF) \& \neg ZF$ // GREATER (signe)
 $\neg(SF \wedge OF)$ // GREATER OR EQUAL (signe)
 $(SF \wedge OF)$ // LESS (signe)
 $(SF \wedge OF) \wedge ZF$ // LESS OR EQUAL (signe)
 $\neg(CF \wedge \neg ZF)$ // GREATER
 $\neg CF$ // GREATER OR EQUAL
 CF // less
 $CF \wedge \neg ZF$ // less or equal

- (F : CARRY FLAG
- ZF: ZERO FLAG
- SF: SIGN FLAG
- OF: OVERFLOW FLAG

MAX: pushl Y.EBP // x = 8(Y.EBP)
 movl Y.ESP, Y.EBP // y = 12(Y.EBP)
 subl \$4, Y.ESP
 movl 8(Y.EBP), Y.ECX // resultat a Y.ECX
 cmpl 12(Y.EBP), Y.ECX
 jle else

if: movl 8(Y.EBP), Y.EAX
 jnp endif
 else: movl 12(Y.EBP), Y.EAX
 \Rightarrow endif: movl Y.EBP, Y.ESP
 popl Y.EBP
 ret

- TRADUCCIÓ SENTENCIES:

- IF - THEN - ELSE: EXAMPLE: int MAX(int x, int y) {
| int max;
| if (x > y) max = x;
| else y = max = y;
| return max; }
- DoWhile: EXAMPLE: int cont(char v){
| int i, cont;
| i = 0; cont;
| do { if (v[i] == '2') ++cont;
| | i++;
| } while (v[i] != '!');
| return cont; }
- while: EXAMPLE: int scd(int a, int b){
| while (b != 0){
| | if (a > b) a = a - b;
| | else b = b - a; ;
| | i&bin 2; ; }

contA: pushl Y.EBP ... cmpb '\$2', (Y.ECX, Y.EDX)
 movl Y.ESP, Y.EBP jne endif
 subl \$8, Y.ESP incl Y.EAX // @v=8[Y.EBP]
 movl \$0, ECX // cont endif: incl Y.EDX
 movl \$0, EDX // i incl Y.EDX
 do: movl 8(Y.EBP), ECX jne do
 ...
 scd: pushl Y.EBP ... subl \$2(Y.EBP), Y.EAX
 movl Y.ESP, Y.EBP jnp endif
 while: cmpl \$0, \$2(Y.EBP) else: subl Y.EAX, \$2(Y.EBP)
 jne end incl: jnp while
 movl 8(Y.EBP), Y.EAX end: popl Y.EBP // a = 8[Y.EBP]
 cmpl \$2(Y.EBP), Y.EAX end: popl Y.EBP // b = 12[Y.EBP]
 jle else

- TIPOS DE DADES ESTRUCTURADES:

- VECTORS: ACCÉS $v[i] = @\text{Inici}V + i \cdot \text{size}$. EXEMPLE: $\text{char } A[12], 1B, 12B, @A + i$
 EXEMPLE: $\text{int } v: (\text{int } v[100], \text{int } i)$ | $v: \text{pushl } y.ebp$
 } $\text{return } v[i];$ $\text{movl } y.esp, y.ebp$
 $\text{movl } 8(y.ebp), y.ecx$ // $@V = 8[y.ebp]$
 $\text{movl } 12(y.ebp), y.edx$ // $i = 12[y.ebp]$; @V
 $\text{movl } (y.ecx, y.edx, 4), y.eax$ // $y.edx = y.edx \cdot 4 + y.ecx$

- MATRÍCS: ACCÉS $v[i][j] = @\text{Inici}A + i \cdot (\text{i.num cols} + j) \cdot \text{size}$. EXEMPLE: $\text{char } *B[80][10], @B + (i \cdot 10 + j) \cdot 4$

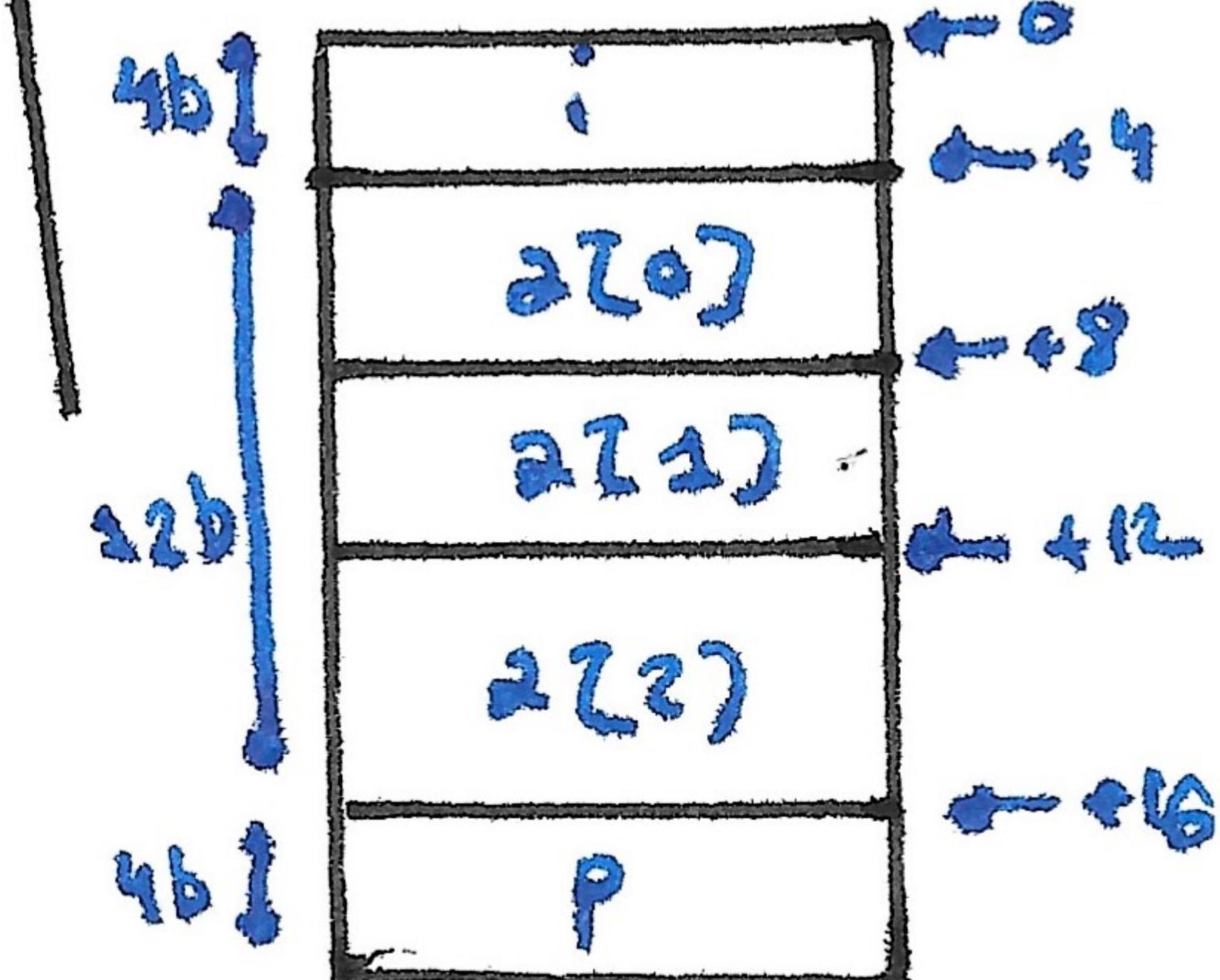
EXEMPLE: $\text{int } Mfc(\text{int } M[50][80], \text{int } fil, \text{int } col) \{$
 } $\text{return } M[fil][col];$

- struct:

EXEMPLE: `typedef struct { int i; int a[3]; int* p; } S;` void Init(S s) {
 } $(s).i = 1;$
 $s.a[2] = 0;$
 $s.p = &(M.a[0]);$

MFC: pushl y.ebp
 movl y.esp, y.esp
 movl 8(y.ebp), y.ecx
 imull \$80, 12(y.ebp), y.edx // $fil = 12[y.ebp] \cdot 80$
 addl 16(y.ebp), y.eax // $col = 16[y.ebp] + fil$
 movl 8(y.ebp), y.ecx // $@M = 8[y.ebp]$
 movl (y.ecx, y.edx, 4), y.eax // $@M + (fil \cdot 10 + col) \cdot 4$

Init: pushl y.ebp
 movl y.esp, y.ebp
 movl \$8(y.ebp), y.edx
 movl \$1, (y.edx)
 movl \$0, 12(y.ebp)
 lsl 4(y.edx), y.eax
 movl y.eax, 16(y.edx)



- subroutines:

- PARÀMETRES: ES PASSEN PER LA PILA DE PRETA A ESQUERRA.

LES MATRÍCS I VECTORES PER REFERÈNCIA, ELS STRUCT PER "

"VAL. TANT chars (1 byte) com shorts (2 bytes) acaben 4 bytes.

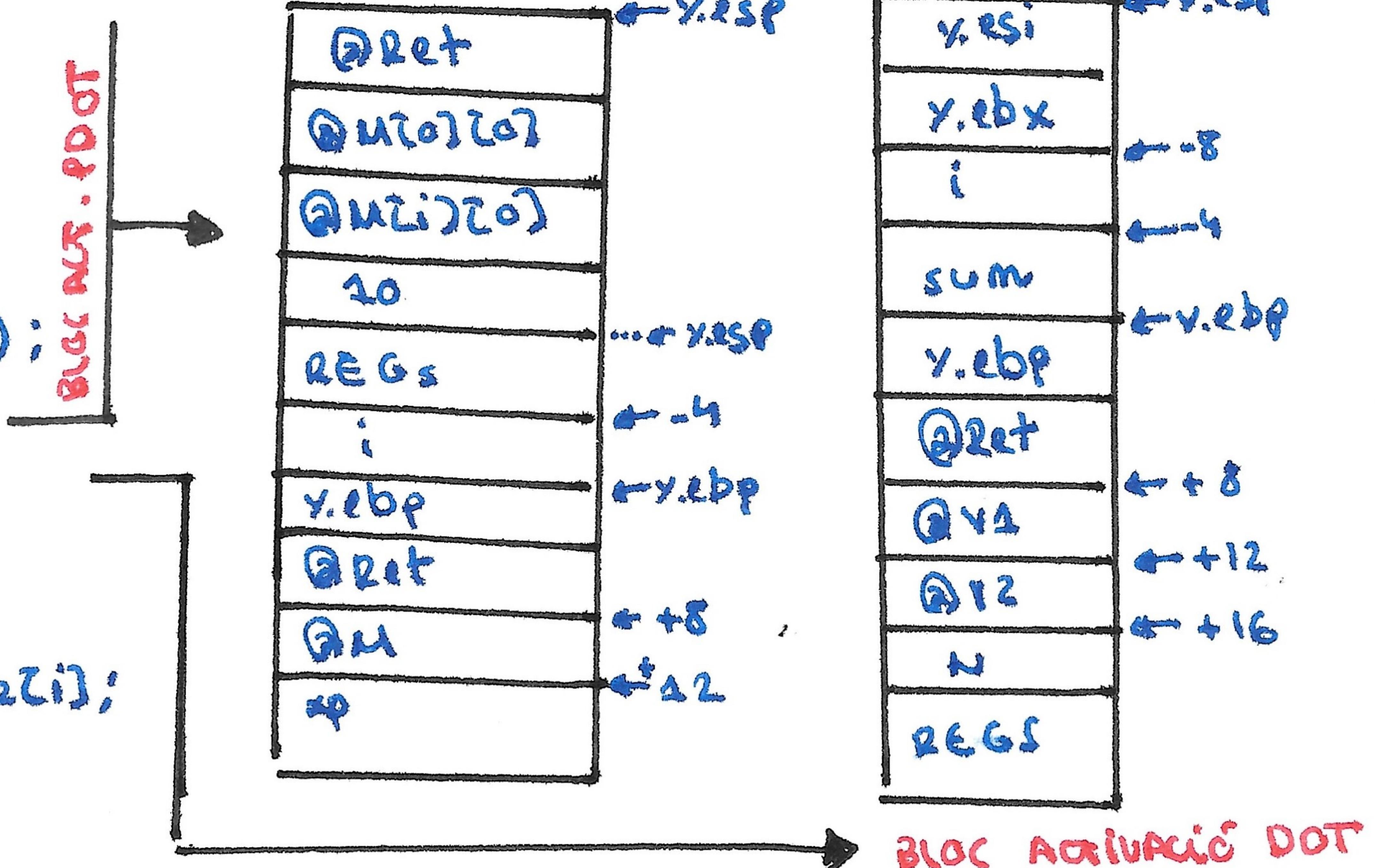
- VARIABLES LOCALES = ALINEADES A LA PILA. char qualsevol direcció, short mult. 2, integers x4.

- REGISTRES: y.ebp, y.esp sempre es SALVEN. y.ebx, y.esi, y.edi SALVAR SI MODIFICATS.

LA PILA SEMPRE ALINEADA A 4.

EXEMPLE: void DOT(int M[10][10], int *p){
 } $\text{int } i;$
 $\text{*p}=0;$
 For($i=0; i<10; i++$)
 $\text{*p+=DOT}(RM[0][0]\&ML[i][0], 10);$
 }

int DOT(int v1[], int v2[], int n){
 } $\text{int } i, sum=0;$
 For($i=0; i<n; i++$) $sum += v_1[i] \cdot v_2[i];$
 return sum;



TEMA 1: FONAMENTS

$$- \text{Potència} = C \cdot V^2 \cdot F , - \text{Potència Fuga} = I_{\text{FUGA}} \cdot V$$

$$- \text{ENERGIA} = C \cdot V^2$$

$$- \text{MTTF} = \lambda^{-1} \text{ TASA FAILS}$$

$$- \text{MTBF} = \frac{1}{\lambda} = \frac{\text{MTTF} \cdot \text{MTTR}}{\text{MTTF} + \text{MTTR}}$$

$$- \text{cost circuit integrat} = \frac{\text{cost die} + \text{cost test} + \text{cost empaquetatse i final test}}{\text{YIELD FINAL}}$$

$$- \text{cost die} = \frac{\text{cost wafer}}{\text{Die Yield} \cdot \text{dies/waffer}}$$

$$- \text{dies/waffer} = \frac{\text{ÀREA ÚTIL}}{\text{die àrea}} = \frac{\pi \cdot (\text{diàmetre}/2)^2}{\text{die àrea}} \cdot \frac{\pi \cdot \text{diàmetre}}{\sqrt{2 \times \text{die àrea}}}$$

$$- \text{die yield} = \text{wafer yield} \cdot \left(1 + \frac{\text{defects per àrea} \cdot \text{die àrea}}{\alpha} \right)^{-\alpha}$$

- MIPS: MILLIONS INSTRUCCIONS PER SEGON.

- MFLOPS: MILLIONS INSTRUCCIONS PER SEGON.

- CYCLES = texe . Ninstr.

$$- \text{AVAILABILITY} = \frac{\text{MTTF}}{\text{MTTF} + \text{MTTR}}$$

$$- P = E / t$$

$$- \text{texe} = \text{tend}^{-1} = \text{Ninst} \cdot \text{CPI} \cdot T_C = \frac{\text{cycles}}{\text{Freq.}}$$

$$- P = I \cdot V$$