Von Neumann (T13)

Única memòria on es emmagatzema les instruccions del programa com les dades.

En cada cicle hi ha una paraula de control diferent (Abans sempre era mateixa).

Recorda que en Harvard l'única és paraula és de la mateixa.

En Multicicle era la mateixa, però hi havia un petit CLS per gestionar mem.

Fases d'execució d'una instrucció

Fetch

Llegeix instrucció de la memòria i la escriu en registre IR.

Aquesta és la primera fase i l'executen TOTES les instruccions.

Decode

"Analitza" la instrucció que hi ha en IR i decideix estat següent.

Això ho determina amb el "Codi d'Operació" i bit "d'extensió"

Fa alguna altre cosa que veurem posteriorment.

Execute

Les accions d'aquesta fase depenen de la instrucció.

Si és de memòria, hi ha una fase addicional.

Fetch

La suma de PC per det. següent instrucció ho realitza la ALU, això es fa en aquesta fase i és guarda en reg del PC amb senyal LdPC # Mantindre PC durant 4 cicles.

Instrucció surt de memòria i es modifica reg IR (senyal LdIr = 1) # També les necessàries per fer que ALU reg PC (senyal LdPC = 1) sumi constant '0x0002'.

Decode

A part de decidir estat següent depenent de la instrucció, en aquest estat es fa:

- Guardar els operands @A i @B en un reg temporal per així ja tindre-los disponibles en pròxim estat i fer l'operació. @A -> Rx, @B -> Ry
- Calcular el pròxim salt (sempre es fa) tot i que resultat no tingui coherència) per si la pròx. instrucció es de salt. El resultat es guarda en reg. temporal. Salt -> R@

⚠ No canvia estat del Computador tot i calcular salt per LdPC = '0'

Ara la ROM que generava les senyals s'anomena "CONTROL UNIT" i és un CLS que determina estat següent. A partir de Instrucció i z genera paraula control.

▽ NO existeix TknBr, ara si es vol "trencar" seq. es fa ficant a '1' LdPc i carregar la nova posició

Obs: "Ld IR" NO és una senyal "sagrada". Quan s'està executant l'estat previ a Fetch, aquesta senyal és 'x' pq. no passa res. El següent estat és Fetch i "machaca" el que hi hagi.

Obs: Quan es fa un BNZ/BZ, tot i que s'ha d'anar al PC de l'ordre que pertoca (depenent el salt), ha de ser -2 pq en F se sumarà +2.

Si has d'anar a PC=0x0004, en F el PC valdrà 0x0002 pq. serà en aquest F que se sumarà +2 fent que en Decode PC = 0x0004.

Obs: Quan es fa "Fetch" Byte = 0 donat que accedim a Memòria, necessitem agafar Word. → Transformar 8b → 16b

▽ R@ ← PC + SE(N8)*2

JALR salta sí o sí Bz = BNz = 1 o LdPc = 1.
#És important que hi hagi un BZ, BNZ previ per saber si saltar o no.
Ra&(~1) Força que Ra sigui parell pq fa AND 1111...1110

Rd guarda el valor del Pc actual.
Ra és el valor de Pc on volem anar

JALR P/I/L/A on ha de ser 11 pq guardem P = PC.

JALR Rd, Ra — Això força a Ra ser parell

$PC = PC+2$; $tmp = Ra \& (\sim 1)$; $Rd = PC$; $PC = tmp$;

Serveix per saltar a una nova posició, però recordant l'origen per poder tornar posteriorment.

0	1	1	1	a	a	a	d	d	d	x	x	x	x	x	x
Codi Op				@A			@D								

▽ Tal i com està muntat tot, no es possible guardar PC en Rd i fer càlcul Ra&(~1) en el mateix cicle donat que tots dos casos requereixen ALU en OP's diff.
El que farem serà afegir nou cable desde PC fins MUX previ a REGFILE
Llavors tindrem complet el senyal "-/I/L/A" → "P/I/L/A"

Altres

Temps de cicle ara és 1400 u.t. → LEU (1350 u.t.) v.s. LDB (1210 u.t)

T_1 (Temps clk=1) = 210 u.t. $T_0 = T_{TOT} - T_1 = 1400 - 210 = 1190$ u.t. = T_0 (senyal clk = 0)

ROM_Q + [01010 0111 x] = 00000 Estem en E10 ≡ JALR → anirem E0 ≡ Fetch
(E10 = 01010, Codi Op = 0111, z = x — #z no fa falta en JALR; E0 = 00000)