ESTRUCTURA DE COMPUTADORS
Grau en Enginyeria Informàtica

Sessió de laboratori número 2

La ruta de dades i control

1. Objectius

- Afermar el funcionament de la ruta de dades del MIPS R2000
- Analitzar com es descodifiquen i executen instruccions de diversos formats.

2. Desenvolupament

En aquesta pràctica treballarem sobre el simulador ProcSim (v2.0) de la ruta de dades del processador MIPS. L'objectiu de la pràctica és familiaritzar-se amb el simulador i comprendre el cicle complet d'execució de diferents instruccions. El simulador s'executa sobre una màquina virtual Java.

3. Presa de contacte amb el simulador: Instruccions de tipus R

Activitat: Iniciar el simulador i carregar una arquitectura i un codi en assemblador.

Per realitzar aquesta pràctica hem d'iniciar el simulador. El simulador està en una carpeta que hem de copiar des de poliformaT en l'escriptori de l'ordinador[1]. Per executar el simulador llancem el programa *ProcSim.exe*. Una volta iniciat, ens apareix una finestra on podem seleccionar la ruta de dades a utilitzar (hi ha diferents versions de la ruta de dades) i el codi assemblador a executar sobre la ruta de dades. El diàleg que ens apareix i les diferents rutes de dades i codi es mostren en les següents figures.

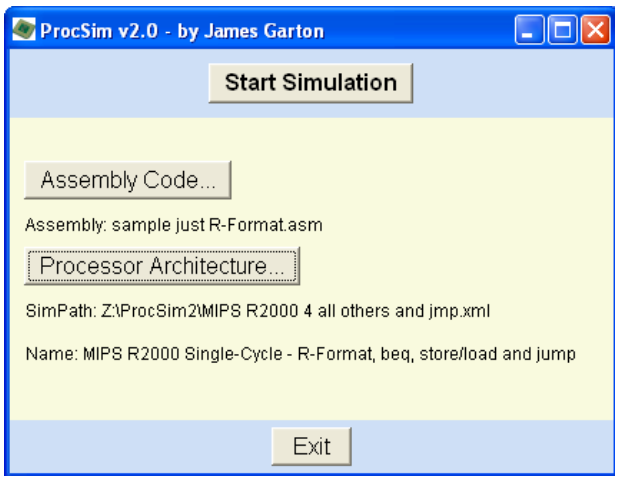


Figura 1. Diàleg principal del simulador.

[1] La carpeta (ProcSim) es pot copiar en qualsevol altra ubicació local de l'ordinador, però, atenció: convé evitar rutes d'accés llargues (ex: C:\MisDocumentos\...\...\ETC\Prac3**ProcSim**), perquè el simulador dona problemes al accedir a les rutes de dades.

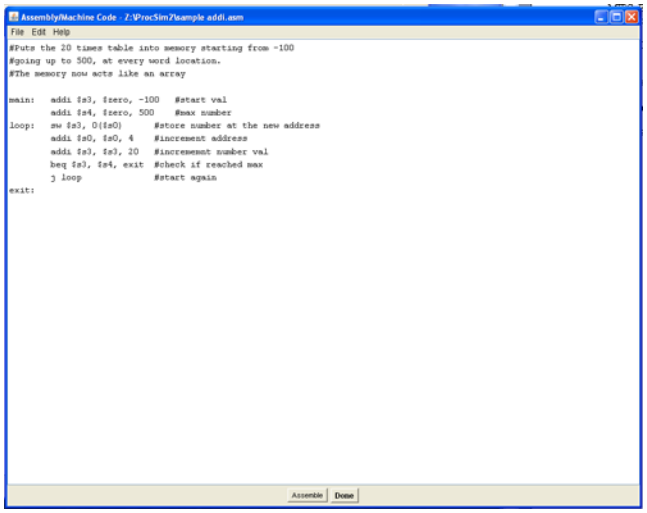


Figura 2. Diàleg d'edició i selecció de codi assemblador.

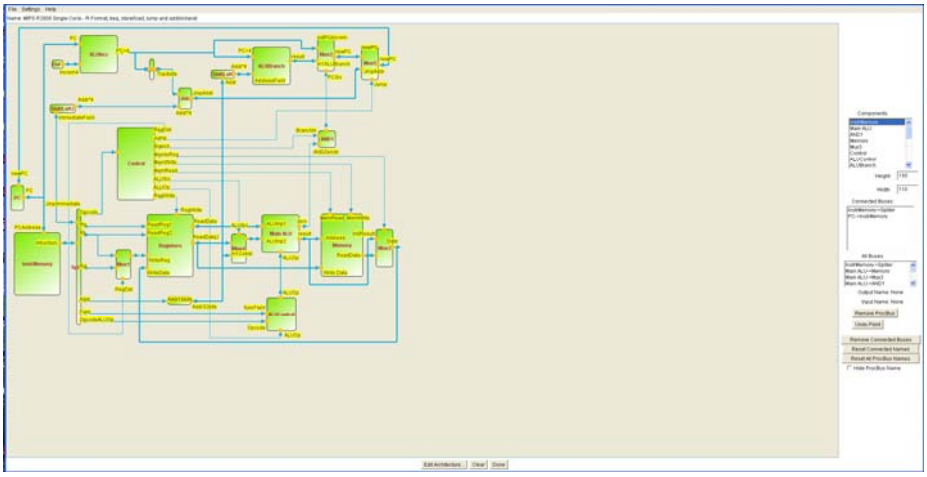
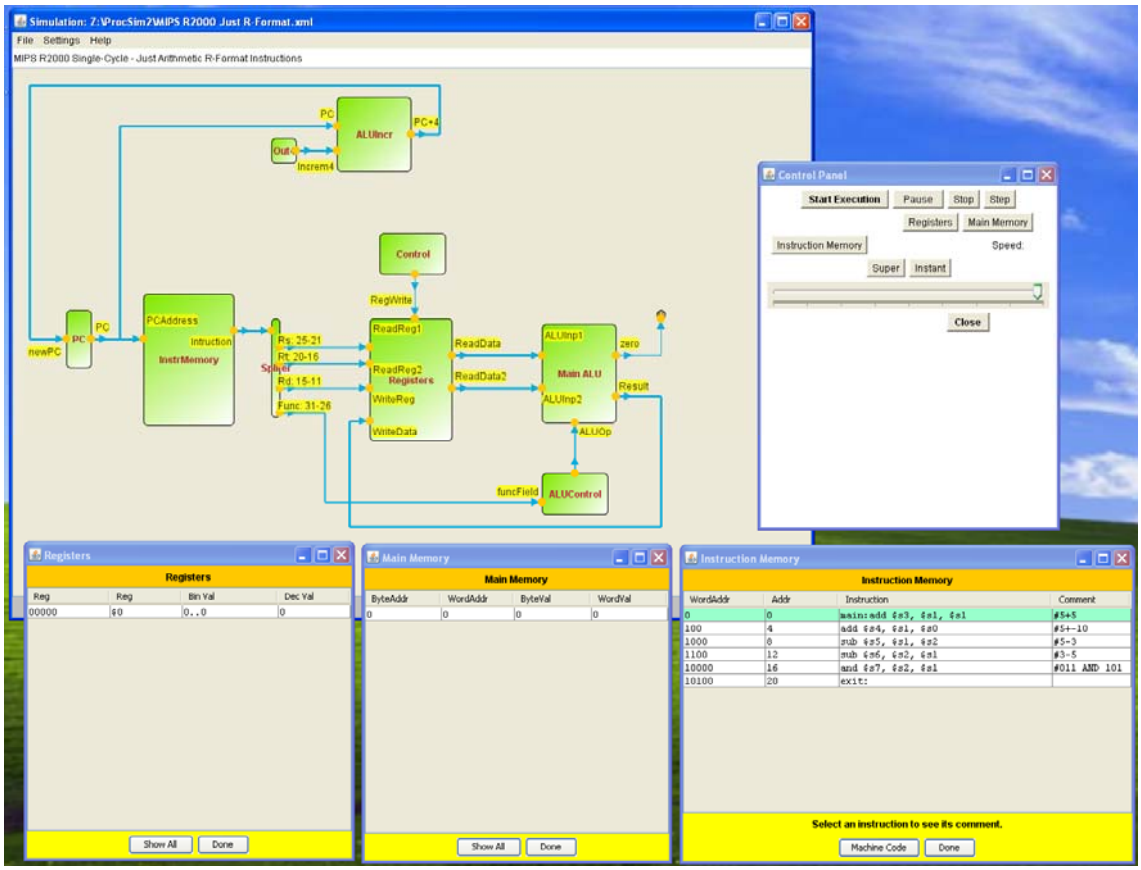


Figura 3. Diàleg de selecció i edició de la ruta de dades a simular.

Les rutes de dades disponibles són subconjunts de la ruta de dades completa vista en les sessions de teoria, encara que alguns dels components reben noms distints.

Selecciona el fitxer de codi assemblador "simple just R-Format.asm", assembla el codi (botó *Assemble*), tanca el quadre de diàleg (botó *Done*) i guarda el canvi (*Save Changes)*. Seguidament, selecciona la ruta de dades "MIPS R2000 Just R-Format" i a continuació tanca la corresponent finestra (botó *Done*). Aquesta ruta de dades mostra els components per a l'execució d'instruccions amb format de tipus R.

Una vegada seleccionats el codi a simular i la ruta de dades sobre la que s'executa, des del diàleg principal podem llançar el simulador fent clic en el botó "*Start Simulation*". Ens apareixen en aquest moment diferents diàlegs, alguns d'ells els podem activar amb els botons en el "*Control Panel*". La següent figura mostra els diferents diàlegs amb la ruta de dades, el banc de registres, la memòria de dades (*Main Memory*), i la memòria d'instruccions amb el codi assemblat.



Activitat: Identifica i familiaritza't amb els diferents components de la ruta de dades. Recorda que la ruta de dades utilitza únicament els components que s'utilitzen al executar instruccions de tipus "R". Els components són:

- Memòria d'instruccions, *InstrMemory*.
- Comptador de programa, *PC*.
- Banc de registres, *Registers*.
- Unitat aritmètic-lògica, *Main ALU*.
- Unitat de control, formada per dos components: Control i ALUControl.
- ALU per a l'autoincrement del comptador de programa, *ALUIncr*.

Tots aquestos components estan interconnectats formant la ruta de dades. En concret tenim connexions entre la Memòria d'instruccions, el comptador de programa i la ALU d'autoincrement per al registre PC. Per altra banda, tenim el banc de registres connectat directament amb la ALU. Identifica en el diàleg de la ruta de dades els diferents components i com es troben interconnectats entre sí.

Una vegada carregat el simulador ja podem començar a executar el codi en assemblador instrucció a instrucció. Per a cada instrucció es visualitza l'avanç de tots els senyals de control i de dades per la ruta de dades. L'execució es controla mitjançant el panell de control (*Control Panel*) amb els botons:

- *Start Execution*. Realitza l'execució de totes les instruccions mostrant de forma animada l'avanç de tots els senyals de control.
- *Pause/Resume*. Realitza una pausa (*pause*) o reprén la simulació (*resume*).
- *Stop*. Finalitza la simulació.

- *Step*. Avança la simulació un pas, representant l'avanç d'un subconjunt de senyals de dades i de control. Per executar una instrucció per complet hem de realitzar alguns passos.
- *Speed*. Podem modificar la velocitat de la simulació amb els botons i la barra associada.

Activitat: Utilitzant el simulador descriu els nou passos en els que s'executa la primera instrucció del codi (`add $s3, $s1, $s1`). Indica per a cada pas quines són las accions realitzades.

Pas	Accions sobre la ruta de dades
1	Fica un 0 al ALUIncr i al Instr Memory
2	Posa l'instrucció add en l'spliter
3	El control el posa a 1
4	rs rt i rd als regitres i func a ALUControl
5	El control envía un 1 als registres i FunControl envia la señal + (010) a Main ALU
6	Llegir els registres el numero 5 i els envia al MainALU
7	Coloca un 10(resultat) en WriteData i trau un 0 en Z perque no és de salt
8	Posa un increment al ALUIncr
9	PC aumenta amb el increment anterior, per a carregar la nova instrucció

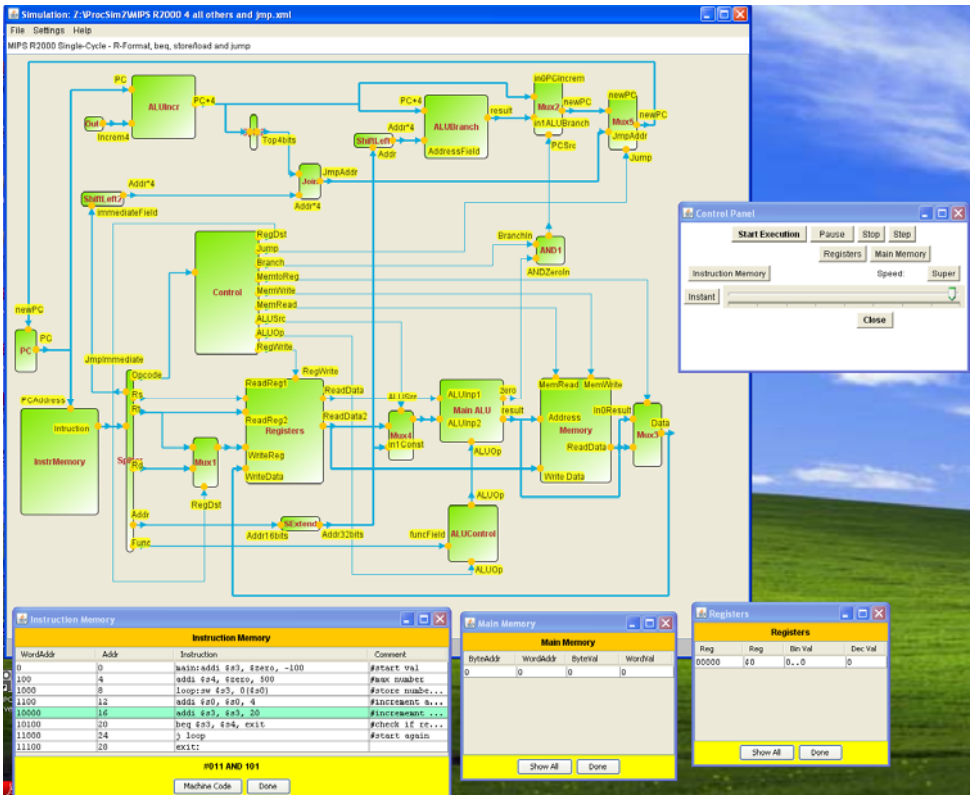
Nota que estem davant una ruta de dades monocicle, en la que totes les accions corresponents a l'execució d'una instrucció tindrien lloc en realitat en el mateix cicle de rellotge. L'ordre en que es mostren les accions en el simulador és arbitrari i simplement pretén indicar de forma llegible l'activació dels senyals de control i el camí que segueixen les instruccions per la ruta.

Activitat: Realitza la simulació completa de les instruccions restants, pas a pas, i identifica en la taula següent els valors dels senyals indicats:

Instrucció	Rs	Rt	Rd	ALUOp	ReadData	ReadData2	Result	Zero
add $s4, $s1, $s0	s1 = 17	s0=16	s4 = 20	010	s1=5	s0=-10	s4=-5	0
sub $s5, $s1, $s2								0
sub $s6, $s2, $s1								0
and $s7, $s2, $s1								0

4. Execució d'instruccions de tipus I i de salt

Una vegada treballada la ruta de dades per a instruccions de tipus R, ara ens centrem en la ruta de dades completa, la qual suporta també instruccions de tipus I e instruccions de tipus J. Per fer-ho, tanca la simulació (botó *Close*) i des del diàleg principal carrega el codi "sample addi" (recorda assemblar el codi amb el botó "*Assemble*") i la ruta de dades "MIPS R2000 5 all jmp and addi". Una vegada carregats fes clic amb el botó "Start Simulation". De nou ens apareixen les finestres amb la ruta de dades (amb més components en aquest cas), el panell de control i els valors emmagatzemats en el banc de registres, la memòria d'instruccions i la memòria de dades. Redimensiona els diàlegs per poder veure tota la informació d'una manera còmoda, tal com es mostra en la següent figura.



En aquesta ruta de dades apareixen molts més components i ara el simulador mostrarà l'execució de cada instrucció en 21 passos. Com elements addicionals tenim:

- Multiplexors:
  - Mux1, multiplexor que selecciona el registre destí en el banc de registres (anomenat MxDST o RegDst en la ruta de dades de teoria).
  - Mux3, multiplexor que selecciona el resultat a escriure en el banc de registres (anomenat MxER o MemToReg en la ruta de dades de teoria).
  - Mux4, selecciona l'entrada de la UAL (anomenat MxUAL2 o ALUSrc en la ruta de dades de teoria).
  - Mux2 i Mux5, multiplexors que seleccionen el nou valor del registre PC (anomenat MxPC o PCMux en la ruta de dades de teoria).
- ALUBranch, calcula l'adreça de salt de les instruccions de salt condicional.
- Operador de desplaçament (Shiftleft), realitza un desplaçament de dos bits a l'esquerra (multiplicació per quatre).
- Operador Join, uneix dos senyals d'entrada, composant un bus que concatena ambdues entrades.
- Operador AND, realitza una operació AND lògica sobre els dos senyals d'entrada.
- L'operador Splitter ja apareixia en l'anterior ruta de dades i serveix per separar els camps d'un bus, en el nostre cas la instrucció actual.

Activitat: Executa les instruccions pas a pas i escriu el valor dels senyals de control (generats per la unitat de control) per a cada instrucció.

Instrucció	RegDst	Jump	Branch	MemtoReg	MemWrite	MemRead	ALUSrc	ALUop	RegWrite
addi $3, $zero, -100									
addi $s4, $zero, 500									
sw $s3, 0($0)									
addi $s0, $s0, 4									
addi $s3, $s3, 20									
beq $s3, $s4, exit									
j loop									

Com es pot observar, en les instruccions de salt condicional (`beq`) i de salt incondicional (`j`), la ruta de dades segueix un camí distint per al càlcul de la nova adreça del PC. En la figura següent podem veure la lògica de la ruta de dades per al càlcul de l'adreça de salt.

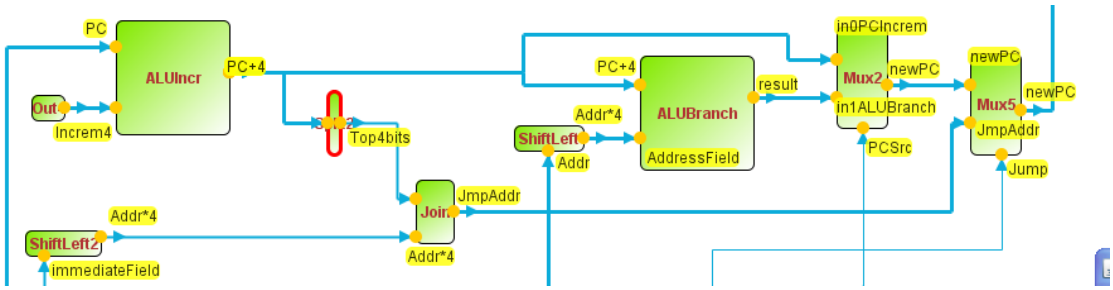


Figura 4. Part de la ruta de dades per al càlcul del nou comptador de programa.

Activitat: Indica els valors d'entrada i eixida per als elements següents, tant per a la instrucció `beq` com per a la instrucció `j`.

Instrucció	Senyals	Valor
Instrucció `beq`	PC+4	
	Addr32bits (eixida SExtend)	
	Addr*4 (eixida shiftleft)	
	result (eixida ALUBranch)	
	PCSrc (eixida AND1)	
	newPC (eixida Mux2)	
	Jump (eixida Mux5)	
	newPC (eixida Mux5)	
Instrucció `j`	JmpImmediate	
	Addr*4 (eixida ShiftLeft2)	
	Top4bits	
	JmpAddr	
	Jump (entrada Mux5)	
	newPC (eixida Mux5)	

Activitat: Justifica el valor dels senyals anteriors, indicant el perquè de eixos valors en funció de l'adreça de salt de les instruccions.