Euskal Herriko Unibertsitatea
Informatika Fakultatea

Konputagailuen Arkitektura eta Teknologia Saila

Konputagailuen Arkitektura I

Kontrol-unitatea 1 (ebazpena): CALL agindua

BIRD prozesadorearen agindu-multzoa zabaldu nahi dugu agindu berri batekin:

```
call azpirrutina
```

Agindu hori azpirrutina bati deitzeko erabiltzen da.

Alde batetik, itzulera helbidea (hau da, PC erregistroaren une horretako balioa) gordetzen da pilaren tontorrean (horretarako, lehenago, pilaren erakuslea, `r31` erregistroa, eguneratu behar da, 1ean inkrementatuz, libre dagoen posizioa erakus dezan); beste aldetik, azpirrutinerako jauzia egiten da PCaren hasierako helbidearekiko modu erlatiboan:

```
r31 := r31+1
MEM[r31] := PC
PC := PC_call + desplazamendua
```

Bere formatua honako hau da:

Idatz ezazu agindu horren exekuzioari dagokion mikroprograma-zatia.
Prozesu-unitatearen egituran aldaketarik egin behar izanez gero, adieraz ezazu argi eta garbi. Beste aginduen kontrolari dagokionez, ba al dago aldaketarik?

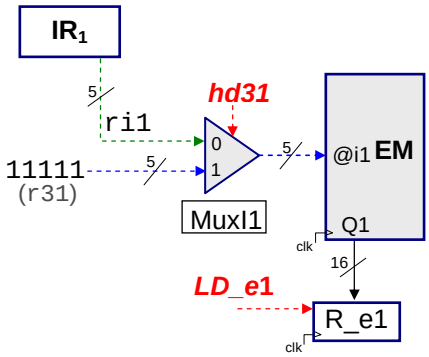
Ebazpena

Batetik, mikroprograma idatzi beharko dugu, eta mikroagindu bakoitzean aktibatu beharreko kontrol-seinaleak adierazi; bestetik, prozesu-unitatean aldaketak egin beharko direnez, zehaztu beharko ditugu.
Mikroprogramari dagokionez, hainbat aukera dauden arren, hurrengoa proposatzen dugu:

```
call1: R_e1 := EM[r31];
call2: R_ual := R_e1 + 1;
call3: MEM[R_ual] := PC; EM[r31] := R_ual; PC := PCi + IR2; goto 0;
```

Hau da, lehendabizi pilaren erakuslea (`r31` erregistroak gordetzen duen helbidea) eskuratu behar da, hurrengo urratsean inkrementatzeko. Beraz, deskodeketa fasearen ondorengo lehen mikroaginduak, `call1` (`R_e1 := EM[r31]`), `r31`an dagoena irakurtzen du, balioa `R_e1` erregistrora kopiatzeko.

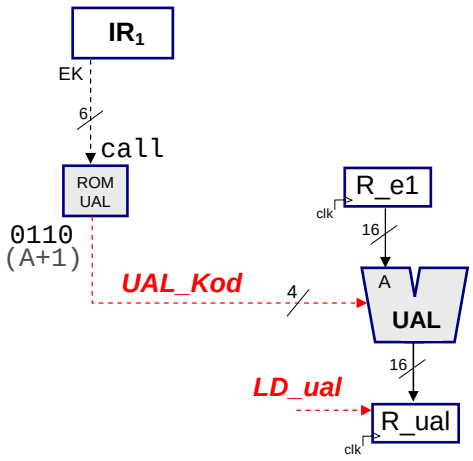
`r31` erregistroa inplizituki erabiltzen denez agindu honetan (ez baita ageri aginduaren formatuan), erregistro-multzoko `@i1` sarrerari adierazi beharko zaio `r31` erregistroa dela irakurri beharrekoa. `@i1` sarreran `IR1` erregistrotik datozen `ri1` izeneko 5 bitak jadanik konektatuta daudenez, BIRD makinaren prozesu-unitatea aldatu beharko da `r31` irakurri ahal izateko. Zein izango da hori egiteko behar den aldaketa? Multiplexore bat (MuxI1) erantsi beharko dugu erregistro-multzoko `@i1` sarreran. Multiplexore horren sarrerak bost bitekoak dira: `IR1` erregistrotik datozen `ri1` bitak, edota bost bateko (31 balioa bitarrez adierazteko, `11111`). Multiplexorearen sarreretako bat aukeratzeko, kontrol-seinale berri bat beharko da, demagun `hd31` izenekoa. `call` agindua denean, bost batekoen segida aukeratu beharko da, adibidez, multiplexorearen 1 sarreran egongo dira bit horiek, eta bestelako aginduetan 0 sarreratik datozen `ri1` bitak aukeratuko dira. Beraz, *`call1`* mikroagindua exekutatu ahal izateko, aldaketa hauek egin behar dira prozesu-unitatean:



Eta *`call1`* mikroagindua exekutatu ahal izateko, kontrol-seinale hauek aktibatu behar dira: `hd31` eta `LD_e1`.

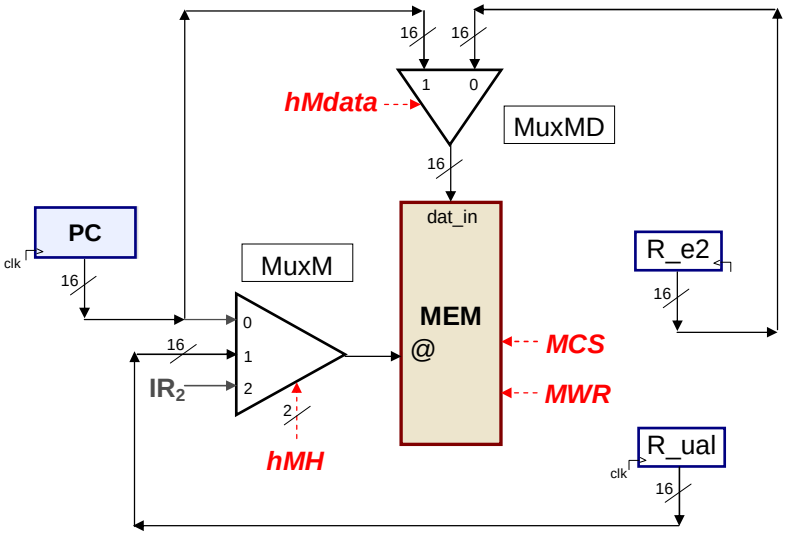
Horren ondoren, *`call2`* mikroagindua (`R_ual := R_e1 + 1`) exekutatu behar da; hala, behin `R_e1` erregistroan kopiatuta dagoela `r31` erregistroaren balioa, inkrementatu behar da, `UAL`a erabiliz. `UAL`ari `ROM_UAL` izeneko memoriatik etorriko zaio inkrementatzeko kodea (`call` aginduaren eragiketa-kodetik abiatuta), eta emaitza `R_ual` erregistroan gordeko da, `LD_ual` kontrol-seinalea aktibatuz.

Hortaz, *`call2`* mikroagindua exekutatu ahal izateko, prozesu-unitatean ez da aldaketa berezirik egin behar, behar diren osagai guztiak jadanik daudelako prozesu-unitatean, baina `ROM_UAL` memoriaren barruan, `call` aginduaren eragiketa-kodeak adierazitako posizioan, `(A+1)` eragiketari dagokion eragiketa-kodea (`0110`) idatzi behar da. Modu horretan, `call` agindua datorrenean, UALak jakingo du inkrementatu behar duela `R_e1` erregistroan dagoena.



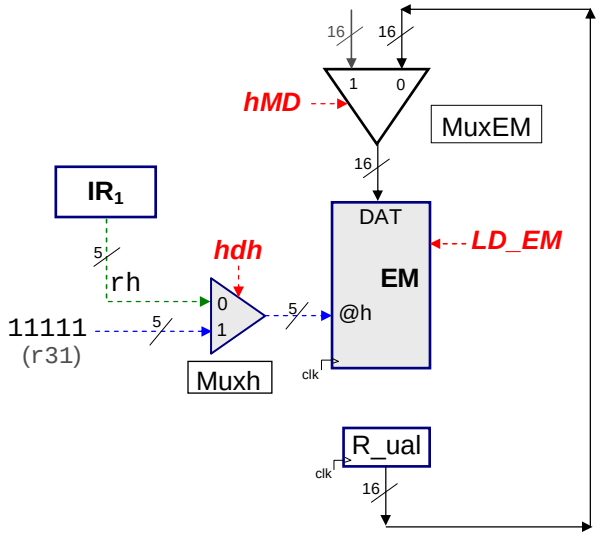
`call3` mikroaginduan eragiketa asko jarri ditugu, baina guztiak paraleloan gerta daitezkeela azpimarratuko dugu (datuek bide ezberdinak segitzen dituztela baliatuz), prozesu-unitatean aldaketa batzuk eginez, berriz ere. Alde batetik, `PC` erregistroaren balioa memorian gorde behar da (`MEM[R_ual] := PC`); bestetik, aurreko urratsean kalkulatutako `r31` erregistroaren balio berria, `R_ual`en dagoena, erregistro-multzoan gorde behar da (`EM[r31] := R_ual`); eta, azkenik, `PC` erregistroari balio berri bat eman behar zaio (`PC := PCi + IR2`) jauzia egiteko. Eragiketa horiek guztiak paraleloan eginda, exekuzioa azkartzen da. Horietako bakoitza exekutatzeko behar dena zehaztuko dugu oraingoan.

Lehenengo eragiketari dagokionez (`MEM[R_ual] := PC`), jatorrizko prozesu-unitatean ez dago modurik `PC` erregistroaren balioa memorian zuzenean gorde ahal izateko, memorian idatz daitekeen datu bakarra `R_e2` erregistrotik datorrena baita. Beraz, `PC` erregistrotik memoriaren `dat_in` sarrerara bidea ireki behar da. Horretarako, `dat_in` sarreran multiplexore bat jarriko dugu orain (MuxMD), eta multiplexorearen 16 biteko sarrerak `R_e2` erregistrotik datorren balioa eta `PC` erregistroak gordetzen duen balioa izango dira. Hemen ere, kontrol-seinale berri bat beharko da erabakitzeko zein den aukeratu beharreko sarrera. Kontrol-seinale berriari `hMdata` deituko diogu eta, adibidez, 0 balioa duenean aukeratuko da `R_e2`tik datorrena eta 1 balioa duenean `PC`tik datorrena aukeratuko da. Hortaz, mikroagindu honetan 1 balioa eman behar zaio `hMdata` seinaleari, eta `MCS` eta `MWR` seinaleak ere aktibatu behar dira memoriako idazketa gauzatu ahal izateko. Idazketa egiteko erabili behar den memoriako helbideari dagokionez, berriz, ez dago inolako arazorik, jatorrizko prozesu-unitatean jadanik dagoelako bidea: hMH = '01' izanik, `R_ual` erregistroaren edukia hartzen da helbidetzat. Hala, prozesu-unitatea honela geratzen da egin beharreko aldaketa sartu ondoren:



Bigarren eragiketari dagokionez (`EM[r31] := R_ual`), agindu honetan `r31` inplizituki erabiltzen denez, erregistro-multzoko `r31` erregistroan balio bat gordetzeko, irakurtzean bezala, `@h` sarreran 31 balioa eman beharko diogu, `IR1`etik datozen 5 bitez gain. Hemen ere, multiplexore bat (Muxh) erantsi beharko da bi sarrerekin (`IR1`etik datorren `rh` bit segida, batetik, eta `11111` bit segida, bestetik) eta kontrol-seinale berri batekin (demagun `hdh` izenekoa). Aurrekoetan egin dugun bezala, multiplexorearen 0 sarreran izango da prozesu-unitatean dagoen jatorrizko balioa, eta 1 sarreran, orain erantsi behar izan duguna (`11111` segida, kasu honetan). Orduan, `R_ual` erregistroan dagoen balioa `r31` erregistroan gorde ahal izateko, `R_ual` erregistrotik erregistro-multzoko datu-sarrerara doan lotura jadanik

egina dagoenez jatorrizko prozesu-unitatean, hMD seinaleak zero balioa izan behar du, eta bakar bakarrik aktibatu behar dira LD_EM eta hdh seinaleak. Aldaketaren ondoren, honela geratzen da prozesu-unitatea:



call3 mikroaginduan egin beharreko azken eguneraketa PC erregistroari dagokiona da (PC := PCi + IR2). Jauzi erlatibo bat egin behar denez, horretarako behar den guztia aurretik dago prozesu-unitatean. Beraz, hD seinalea aktibatuz MuxPCen PCi aukeratzen da, eta MuxBaten IR2an dagoen desplazamendua. Bi horiek batzen dira PCaren sarreran dagoen batugailuan, eta LD_PC kontrol-seinalea aktibatzean balio hori gordetzen da PC erregistroan.
Hortaz, laburbilduz, *call3* mikroaginduan aktibatu beharreko kontrol-seinaleak hMdata, MCS, MWR, hdh, LD_EM, hD eta LD_PC dira, eta hMHren balioa 01 izan behar da.

Sekuentziazioari dagokionez, *call1* mikroaginduaren ondoren exekutatu beharrekoa beti *call2* mikroagindua izango da, eta *call2* mikroaginduaren ondoren, *call3* mikroagindua izango da beti. Beraz, bi kasu horietan baldintza kodea 00 da (zero konstanteari dagokiona, ez baita jauzirik egin behar sekula), deskodeketako bita 0 da, eta jauzi helbideko bost bitetan berdin zaigu zer jarri, ez baita sekula jauzirik egingo. *call3* mikroaginduaren kasuan, ordea, ondoren beti 0 mikroagindua exekutatu behar denez (goto 0), baldintza kodea 11 da (bateko konstantea, beti jauzia egin behar delako), deskodeketa bita 0 izango da, eta jauzi helbideko bost bitak zeroak izango dira, hurrengo aginduaren lehen bilaketa fasera joan ahal izateko. Mikroaginduak bitarrean nola geratzen diren jarraian adierazten dugu.

egoera	BK1BK0	Desk	S4S3S2S1S0	mcs	moe	mwr	hMH	ld_IR1	ld_IR2	ld_EM	hMD	hH	ld_e1	ld_e2	hB	ld_ual	ld_PC	ld_PCi	hD	hd31	hMdata	hdh
call1	00	0	xxxxx	0	0	0	xx	0	0	0	x	x	1	0	x	0	0	0	0	1	x	x
call2	00	0	xxxxx	0	0	0	xx	0	0	0	x	x	0	0	x	1	0	0	0	x	x	x
call3	11	0	00000	1	0	1	01	0	0	1	0	x	0	0	x	0	1	0	1	x	1	1

Argi dago, kontrol-seinale berriak sartu behar izan ditugunez, erantsi beharreko aldaketak zirela eta, mikroaginduen luzera handitu dela. Beste aginduen exekuzioan, gehitu ditugun kontrol-seinale berriak desaktibatuta egongo dira, ariketan zehar aipatu diren kasuetan salbu.