Estructura bàsica d'un computador

Montse Peiron Guàrdia
Fermín Sánchez Carracedo

Índex

Introducció

Aquest apunts són una guia per a l'estudi del llibre o dels apunts que descriuen la Màquina Rudimentària.

En aquest tema farem servir els coneixements adquirits en els temes anteriors per a aprendre com estan formats els computadors a nivell electrònic. L'aprenentatge es farà sobre un computador pedagògic, la Màquina Rudimentària, que té la mateixa estructura bàsica que la majoria dels ordinadors reals.

Segons aquesta estructura, definida per John von Neumann, el computador es divideix en tres parts clarament definides:

- La memòria
- La Unitat Central de Procés (UCP o CPU)
- La unitat d'entrada/sortida

Els ordinadors treballen amb bits. Per tant, qualsevol informació que hagi d'ésser emmagatzemada en un ordinador ha d'ésser traduïda a una cadena de bits. Existeixen moltes codificacions estàndard per a representar diferents tipus de dades, tal com s'ha vist en el tema "Representació de la informació". Generalment, els nombres naturals es codifiquen en *binari pur*, mentre que els nombres enters es codifiquen en *complement a dos*.

Els programes també s'han de codificar en bits perquè el computador els pugui entendre. Aquesta és la funció del *llenguatge màquina*. Atès que els bits són difícilment intel·ligibles per a les persones, hom ha ideat el *llenguatge assemblador*, que és una traducció mes o menys directa del llenguatge màquina però empra lletres i nombres.

Les instruccions d'un programa indiquen al computador quines operacions ha de realitzar, i també sobre quines dades ha de fer els càlculs. Els *modes d'adreçament* permeten indicar a la màquina on hi ha emmagatzemades aquestes dades.

La part de l'ordinador encarregada d'executar els programes és la Unitat Central de Procés. Es divideix en la *Unitat de Procés*, que és on es duen a terme les operacions, i la *Unitat de Control*, que és un sistema seqüencial encarregat de governar el funcionament de la Unitat de Procés.

Aquests apunts es centren en l'estudi de tots aquest conceptes i en la seva aplicació a un computador senzill: la Màquina Rudimentària.

Objectius

Aquest tema permetrà a l'estudiant adquirir els coneixements següents i aprofundir-hi:

1. Estructura d'un computador von Neumann
2. Llenguatges màquina i assemblador
3. Codificació d'instruccions
4. Formes d'accedir als diferents espais d'adreces d'un computador
5. Arquitectura d'un computador pedagògic: la Màquina Rudimentària

Per a entendre bé aquest tema cal tenir assimilades algunes nocions bàsiques de programació. Es recomana a aquells estudiants que no en tingueu cap coneixement i no hàgiu cursat l'assignatura "*Introducció a la programació "* o l'estigueu cursant, que us mireu l'Apèndix B d'aquests apunts, on s'expliquen els conceptes necessaris.

El text on s'expliquen els conceptes corresponents a aquest tema són els capítols 6 i 7 i l'apèndix del llibre *Fundamentos de Computadores*. Aquests apunts constitueixen una guia per a l'estudi del text. En concret, els apunts contenen una introducció als diferents apartats del llibre i explicacions addicionals que en faciliten la comprensió.

L'estudiant pot analitzar per si mateix si ha entès correctament la matèria mitjançant l'ús d'un simulador de la Màquina Rudimentària. Aquest simulador és de lliure distribució i el podeu carregar des d'aquesta adreça de ftp de la xarxa:

ftp://ftp.ac.upc.es/pub/archives/mr/win31/mr_20.zip

D'altra banda, el primer apartat dels apunts conté un repàs de la codificació de nombres enters en complement a 2, que ja s'ha vist en el tema "Representació de la informació" d'aquesta assignatura.

La manera òptima per a estudiar el tema és, per a cada apartat, llegir en primer lloc els apunts i després aprofundir en els detalls que es faciliten en el llibre o en els apunts de l'MR. Els apunts contenen algunes orientacions respecte a l'ordre en què cal estudiar els diferents conceptes. Totes les indicacions estan ombrejades en gris. És convenient que aneu fent els problemes a mesura que aneu avançant als apunts.

1. Repàs: representació de nombres enters en complement a dos

La Màquina Rudimentària treballa amb nombres enters de setze bits codificats en complement a dos (Ca2). Per això, en aquest apartat farem un breu resum de com funciona aquesta representació, que ja va ésser estudiada en el tema "Representació de la informació".

Amb n bits es poden representar 2^n combinacions diferents. Per tant, el rang de nombres que es poden representar en Ca2 depèn del nombre de bits de què es disposi. Amb n bits es pot representar des del nombre -2^{n-1} fins al nombre $2^{n-1} - 1$.

Donat un nombre codificat en Ca2 amb n bits, la forma de saber quin nombre enter x representa és la següent. Suposem que Xe és el valor enter del nombre si l'interpretem en binari pur.

- Si el bit de més pes és 0, llavors el nombre és positiu, i el seu valor és el mateix que si estigués codificat en binari pur; és a dir, $x = X\text{e}$.
- Si el bit de més pes és 1, llavors el nombre és negatiu, i el seu valor és $x = X\text{e} - 2^n$.

El bit de més pes d'un nombre codificat en Complement a 2 indica el signe del nombre.

La regla per a saber com ha de representar-se en Complement a 2 i n bits un nombre enter x és la següent:

- Si x és positiu, es codifica igual que en binari pur;
- Si x és negatiu, s'ha de codificar en binari pur el nombre $2^n + x$

Per exemple, amb 4 bits:
- El nombre positiu 1 es representa igual que en binari pur, és a dir, 0001.
- El nombre negatiu –1 es representa com el $2^n + (-1) = 2^4 - 1 = 15$ en binari pur; és a dir, 1111.

La manera més senzilla de canviar de signe un número representat en complement a dos consisteix en canviar zeros per uns i viceversa i sumar 1 al resultat.

Per exemple, amb 4 bits el 3 es representa en Ca2 com a 0011. Després de canviar-ne el signe s'obté el nombre 1101 (1100+1), que és la representació en Ca2 del nombre –3.

2. Arquitectura bàsica d'un computador

2.1. Jerarquia de nivells d'un computador

Un computador es pot veure de maneres molt diferents: com un conjunt de circuits electrònics, o com una màquina que ens permet escriure i guardar textos, per exemple. De la mateixa manera que un cotxe és tractat a diferents nivells per un enginyer, un mecànic, un venedor, un conductor o un passatger, un computador també té diferents nivells.

Després de llegir aquest apartat heu de tenir una idea de la funció de cadascun dels nivells. No cal estudiar-lo en profunditat, només a nivell de divulgació.

Aquest curs se centra en l'estudi dels dos nivells inferiors: el nivell de circuits digitals i el nivell de llenguatge màquina.

2.2. Computadors von Neumann

Aquest tema se centra en l'estudi bàsic dels dos nivells inferiors: lògica digital i llenguatge màquina, que són els relacionats amb el maquinari del computador.

A aquest nivell, la majoria dels computadors comparteixen la mateixa *estructura lògica* o *estructura*, és a dir, estan formats pel mateix conjunt d'unitats, cadascuna de les quals té una funció definida, i les unitats s'interrelacionen entre elles d'acord amb el mateix patró.

John von Neumann i altres van definir l'any 1946 aquesta estructura lògica comuna, que consisteix en tres grans unitats:

- **Memòria principal**: emmagatzema els programes, que estan formats per instruccions (ordres) i dades sobre les quals han de treballar les instruccions. També guarda les dades que s'obtinguin com a resultat de l'execució dels programes.

- **Unitat Central de Procés** o **processador**: és on s'executen els programes, és a dir, on es duen a terme els càlculs i accions indicades en els programes. Té dues grans subunitats:

    - la **Unitat de Procés** és el conjunt de circuits electrònics necessaris per a l'execució dels càlculs dels programes.

    - la **Unitat de Control** és un sistema seqüencial que governa la Unitat de Procés, fent que la seqüència temporal d'esdeveniments sigui l'adequada per a la correcta execució dels programes.

- **Unitat d'Entrada/Sortida**: connecta el computador amb l'usuari a través dels perifèrics, ja siguin d'emmagatzematge (discos, cintes, etc.) o de comunicació amb l'usuari (impressores, mòdems, teclats, pantalles, etc.)

Hom acostuma a anomenar *CPU* el processador, sigles de les paraules angleses *Central Processing Unit*.

Aquestes unitats estan interconnectades mitjançant un conjunt de busos:

- El **bus de control**: transporta senyals de control entre el processador i la resta d'unitats
- El **bus de dades**: transporta dades entre les diferents unitats del computador
- El **bus d'adreces**: transporta adreces des del processador fins a les altres unitats

Funcionament de l'ordinador von Neumann

A grans trets, per a executar els programes se segueix el procés següent.

Els programes estan guardats en un disc (ja sigui el disc dur, o compacte –CD–, o un disc flexible). La unitat d'Entrada/Sortida porta el programa a executar des del disc fins a la memòria principal, i a continuació s'executen les instruccions del programa en el processador una per una.

L'execució de cada instrucció consta dels passos o fases següents:

1. Es porta la instrucció des de la memòria fins al processador (fase de ***fetch***)
2. El processador examina quines són les ordres que especifica la instrucció (fase de ***descodificació***)
3. El processador localitza les dades (***operands***) que requereix la instrucció (fase de ***lectura d'operands***)
4. El processador fa els càlculs indicats per la instrucció (fase **d'*execució***)
5. Es guarda (si cal) el resultat d'aquests càlculs (fase **d'*escriptura de resultats***)

Una vegada completada l'execució d'una instrucció es repeteixen aquests passos per a la propera instrucció que s'hagi d'executar, i així successivament fins a completar l'execució de tot el programa.

En principi les instruccions s'executen en seqüència, en l'ordre en què estan emmagatzemades a la memòria. El registre *PC* (comptador de programa) indica en tot moment la posició de memòria on hi ha la propera instrucció que s'ha d'executar. Les instruccions anomenades *de **salt*** poden trencar aquest ordre d'execució.

En aquest punt cal...

- Conèixer la funció de cadascuna de les unitats d'un computador von Neumann.
- Saber quins busos connecten aquestes unitats i el tipus d'informació que transporten.
- Tenir una idea de la missió dels components més importants de la Unitat de Procés.
- Entendre el procés d'execució dels programes.

2.3. Descripció de l'arquitectura de la Màquina Rudimentària

La Màquina Rudimentària (MR) és un computador molt senzill, d'ús fonamentalment didàctic, que serveix per a il·lustrar tots els conceptes sobre estructura de computadors que s'expliquen en aquests apunts.

Les característiques concretes de les diferents unitats de la MR són aquestes:

- Memòria principal: 256 mots de 16 bits. Guarda instruccions i dades, que són nombres enters codificats en 16 bits i complement a 2.

- Unitat Central de Procés: Conté un banc de registres amb 8 registres de 16 bits, un registre comptador de programa (PC) de 8 bits, un registre de instruccions (IR) de 16 bits, dos indicadors de condició (Z i N) i una UAL que pot fer les operacions de sumar, restar, desplaçar un bit a la dreta i *AND* lògica.

- Sistema d'Entrada/Sortida: la MR no té unitat d'Entrada/Sortida, perquè l'estudi d'aquest sistema es farà a l'assignatura "Estructura de Computadors I". Per tant, assumirem que els programes ja estan carregats a la memòria, sense preguntar-nos com hi han anat a parar.

- Busos d'interconnexió: el bus de dades és de 32 bits (16 bits de la memòria al processador i 16 bits en sentit contrari), el d'adreces, de 8 bits, i el de control, d'1 bit.

En aquest punt cal...

- Aprendre les característiques concretes de tots els components de la MR que s'esmenten en aquest apartat.
- Entendre la utilitat d'aquests components.
- Saber justificar l'amplada de tots els busos.

2.4. Nivell de llenguatge màquina de la Màquina Rudimentària

Llenguatges d'alt nivell i de baix nivell

Els ordinadors estan formats per circuits digitals, i per tant només entenen dos símbols, el 0 i l'1. En conseqüència, tota la informació que manegin ha d'estar codificada en bits.

Normalment, els programadors d'ordinadors escriuen els programes en algun llenguatge de programació (Fortran, C++, etc.) que és més o menys proper al llenguatge natural. Per tal que l'ordinador pugui executar el programa, aquest ha d'ésser traduït a un llenguatge que contingui bits i prou: el **llenguatge màquina**. L'encarregat de fer la traducció és el **compilador**.

El llenguatge màquina consisteix en un conjunt d'instruccions que poden ésser codificades en bits segons unes regles de traducció. Així, un programa en llenguatge màquina és una cadena de 0 (zeros) i 1 (uns). A les persones ens resulta incòmode desxifrar aquestes cadenes de bits, i per això hom ha ideat el **llenguatge assemblador**. Cada instrucció de llenguatge assemblador correspon a una instrucció en llenguatge màquina. La diferència entre ambdues és que la primera està codificada en bits i la segona en lletres i nombres, d'una manera que resulta fàcilment intel·ligible a les persones.

En llenguatge assemblador, els noms de les instruccions són mnemotècnics que deriven de paraules en anglès, de manera que ens resulta molt fàcil associar cada instrucció amb la funció que fa.
Per exemple, ADD són les primeres lletres de la paraula anglesa *addition*.

Per exemple, una de les instruccions que forma part de tots els ordinadors és la que dóna ordre de realitzar una suma. Una possible codificació en llenguatge màquina podria ser 0110, mentre que en llenguatge assemblador sol anomenar-se ADD. Clarament, la segona opció ens resulta més entenedora.

Els llenguatges màquina i assemblador s'anomenen **llenguatges de baix nivell**, en contraposició als que usem normalment per a programar, que s'anomenen **llenguatges d'alt nivell**. La diferència és que en un llenguatge d'alt nivell el programador especifica quins càlculs vol fer, i en un llenguatge de baix nivell es detallen les operacions que ha de dur a terme el computador per tal de realitzar-los (per això es diu que els llenguatges d'alt nivell "tenen un contingut semàntic més elevat").

Per exemple, la instrucció d'alt nivell A:= B + C indica que s'ha de sumar el valor de les variables B i C i guardar el resultat a la variable A, però no explicita quines posicions de memòria ocupen les variables ni com s'ha de dur a terme l'operació.

Imaginem que les variables A, B i C es guarden, durant l'execució del programa, a les posicions de memòria 8850h, 8851h i 8852h respectivament. En ser traduïda a baix nivell, la instrucció A:= B + C es transforma en una seqüència d'instruccions com aquesta:

```
MOV 8851h, R1    ; copiar el valor de la variable B sobre el registre R1
MOV 8852h, R2    ; copiar el valor de la variable C sobre el registre R2
ADD R1,R2, R3    ; sumar els registres R1 i R2 i deixar el resultat al registre R3
MOV R3, 8850h    ; copiar el valor del registre R3 sobre la variable A
```

Molts computadors només fan operacions amb dades contingudes en els registres de la Unitat de Procés. Per tant, les variables s'han de copiar sobre registres abans de fer les operacions.

Codificació d'operacions i operands

Una instrucció de baix nivell dóna tres tipus d'informacions al computador:

- Quina és l'**operació** que ha de realitzar.
- Sobre quines dades, o **operands**, ha de fer l'operació.
- On s'ha de guardar el **resultat** (si n'hi ha)

Per a donar aquestes informacions, les instruccions s'estructuren en camps. Cadascun dels camps de les instruccions pot ésser de longitud fixa o variable.

El camp **codi d'operació** indica l'operació que cal realitzar. En llenguatge assemblador correspon al nom de la instrucció, que és un mnemotècnic que ens recorda l'operació que s'ha de fer. En l'exemple anterior, la instrucció que ordena fer un *moviment* d'una dada entre diferents parts del computador es diu *MOV*, i la instrucció que ordena fer una *suma* es diu *ADD*.

La longitud del camp **codi d'operació** determina el nombre d'instruccions de llenguatge màquina del computador. Per exemple, si el camp codi d'operació és de longitud fixa de 8 bits, es poden codificar fins a $2^8 = 256$ instruccions diferents.

Hi ha dos tipus d'operands: **operands font** (les dades sobre les que es fan les operacions) i **operands destí** (el lloc on es guarda el resultat). Hi ha un camp diferent per a codificar cadascun dels operands de la instrucció. Aquests camps indiquen la manera d'accedir a l'operand mitjançant els **modes d'adreçament**. En l'exemple anterior, les instruccions de moviment tenen dos operands (un font i un destí), mentre que la de suma en té tres (dos fonts i un destí).

Modes d'adreçament

Els modes d'adreçament són el mecanisme que permet a les instruccions de llenguatge màquina indicar al computador on estan situades les dades necessàries per a la seva execució.

Per a localitzar una dada cal indicar:

- l'espai d'adreces on està situada,
- el lloc que ocupa dins d'aquest espai d'adreces.

En aquest apartat es descriuen els quatre modes d'adreçament més habituals dels computadors, que són els que usa la Màquina Rudimentària. En l'assignatura "Estructura de Computadors I" s'estudien altres modes d'adreçament.

Espais d'adreces

S'anomenen **espais d'adreces** els llocs on poden estar emmagatzemades les dades que utilitza un programa.

En aquest curs distingirem tres espais d'adreces diferents:

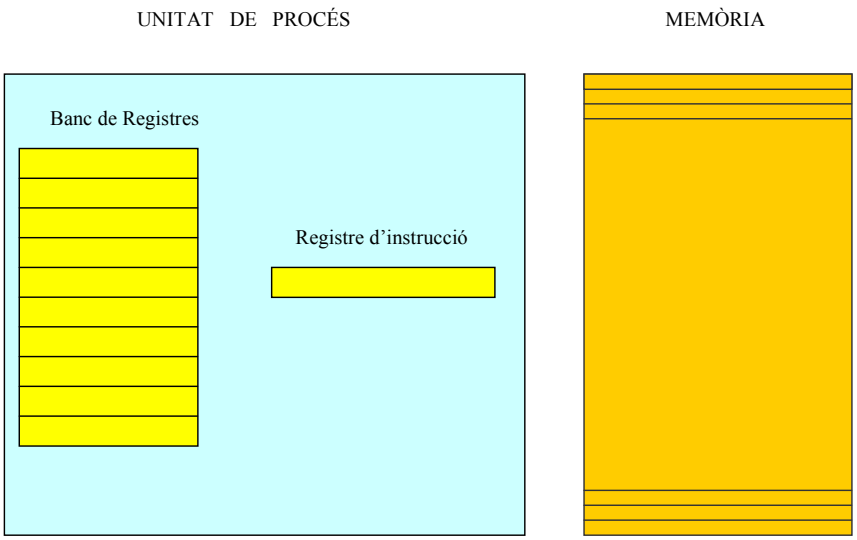
- La *memòria*: conté els programes i les dades. El seu temps d'accés és elevat. Hi ha diversos nivells de memòria, tal com s'estudiarà en les assignatures "Estructura de Computadors I" i "Estructura de Computadors II".

- El *banc de registres de la Unitat de Procés*: és una unitat d'emmagatzematge amb un temps d'accés més reduït que el de la memòria. Normalment no té gaires registres (entre 32 i 128), i per tant és important fer-ne un bon ús. Durant l'execució dels programes, guarda les dades que arriben de la memòria i altres valors temporals necessaris per a la seva execució.

Els registres de la Unitat de Procés també s'anomenen registres *de propòsit general*, per a distingir-los dels registres que tenen funcions específiques (com ara el Comptador de Programa o PC).

- El *registre d'instrucció*: És un registre especial de la Unitat de Procés que conté en cada moment la instrucció que s'està executant. És pot considerar un espai d'adreces perquè en algunes ocasions les dades estan contingudes dintre de la pròpia instrucció.

La figura següent mostra els tres espais d'adreces d'un computador que s'estudien en aquest curs.



Estudiarem a continuació les formes d'indicar a una instrucció quin és l'espai d'adreces on es troba un operand i com accedir-ne.

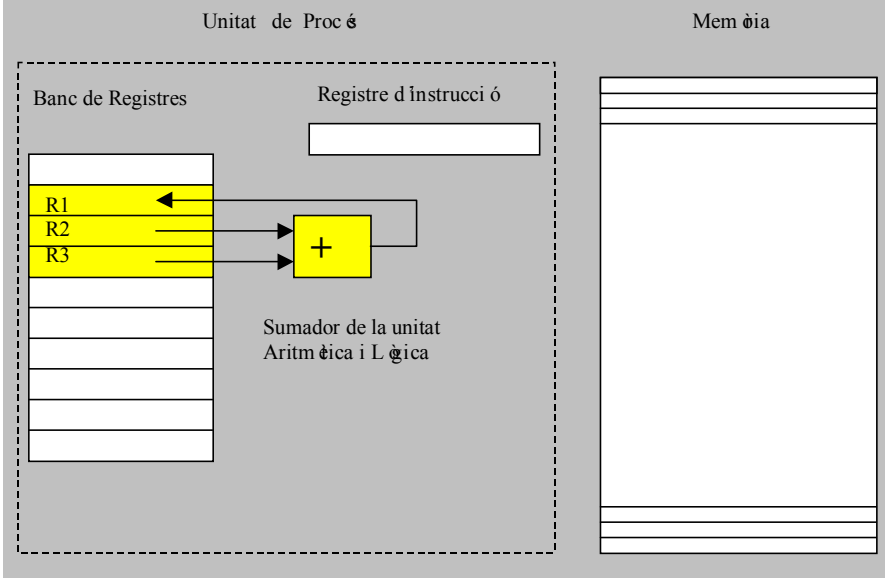
Mode directe a registre

Aquest mode permet d'accedir a un operand que està emmagatzemat en qualsevol dels registres del Banc de Registres de la Unitat de Procés.

Per a especificar on és l'operand s'escriu directament el nom del registre que el conté. Normalment, els registres tenen un nom que identifica el tipus de dada que contenen i la seva posició al Banc de Registres. Aquest nom és, en molts computadors, una lletra seguida d'un nombre. La lletra "R" acostuma a indicar que es tracta d'un registre que conté nombres enters, mentre que la lletra "F" s'usa per a nombres reals escrits en coma flotant. Els registres estan agrupats en bancs de registres de tipus homogeni.

Per exemple, si una instrucció especifica R4 com un dels seus operands, està indicant que l'operand és al registre número 4 del Banc de Registres d'enters. Si l'operand fos F4, es referiria al registre 4 del Banc de Registres de nombres reals.

Per exemple: la instrucció *ADD R2,R3,R1* escriu en el registre R1 la suma dels continguts del registres R2 i R3. L'execució d'aquesta instrucció es mostra en la figura següent.



La Màquina Rudimentària treballa amb un sol tipus d'operands: nombres enters de 16 bits codificats en complement a dos, i per tant té un sol Banc de Registres.

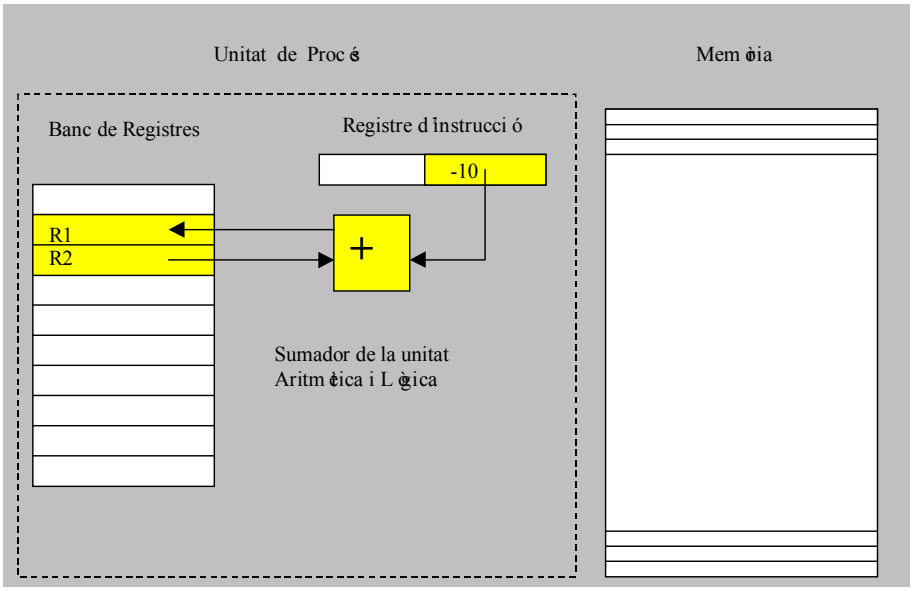
Mode immediat

Aquest mode permet d'especificar un operand que està codificat dins la pròpia instrucció.

Per a especificar l'operand s'escriu el valor d'aquest operand precedit (habitualment) del símbol "#". El valor es pot escriure en base decimal o en altres bases (en aquest cas cal indicar-ho, tal com es veurà més endavant). Aquest tipus d'operand es denomina operand immediat.

Aquest mode d'adreçament no pot ésser mai usat com a operand destí d'una instrucció.

Per exemple, la instrucció *ADD R2,#–10,R1* escriu en el registre R1 la suma del contingut del registre R2 amb el valor –10. En la figura següent es mostra l'execució d'aquesta instrucció.



Mode absolut

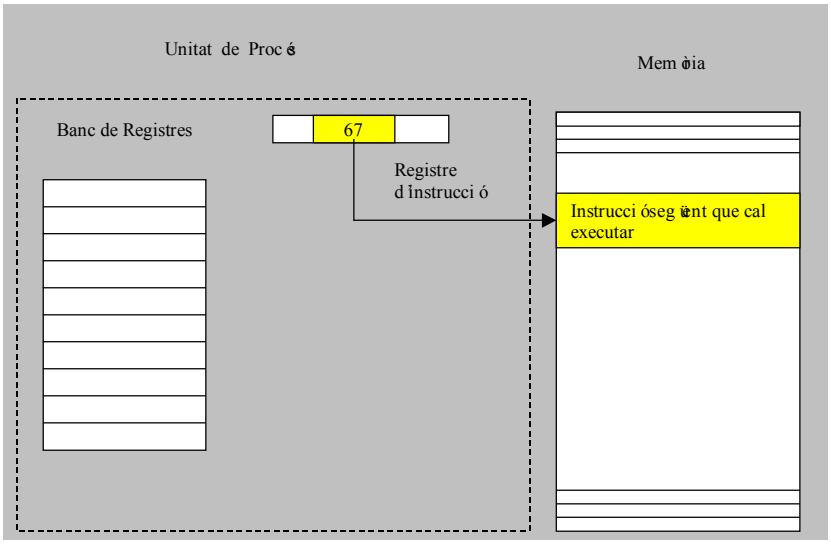
Aquest mode permet d'especificar un operand que està guardat en una posició de memòria.

Per a indicar on és l'operand s'escriu l'adreça de la posició de memòria que el conté.

En la Màquina Rudimentària, aquest mode es fa servir només en les *instruccions de salt*, en les quals l'operand és la propera instrucció a executar. Les instruccions s'executen normalment en l'ordre en què estan emmagatzemades a la memòria, és a dir, després de la instrucció que hi ha a l'adreça X s'executa la instrucció que hi ha a l'adreça $X + 1$. Però, si la instrucció que hi ha a l'adreça X és una instrucció de salt, com per exemple *BR 67* (saltar a l'adreça 67), la següent instrucció a executar serà la que hi ha a l'adreça 67.

El nombre de bits dedicats a codificar un operand en mode absolut depèn de la grandària de la memòria principal. En general, si la memòria té una capacitat de 2^k *bytes*, es codificarà amb k bits.

En la figura següent mostrem la posició de l'operand en la instrucció *BR 67*.

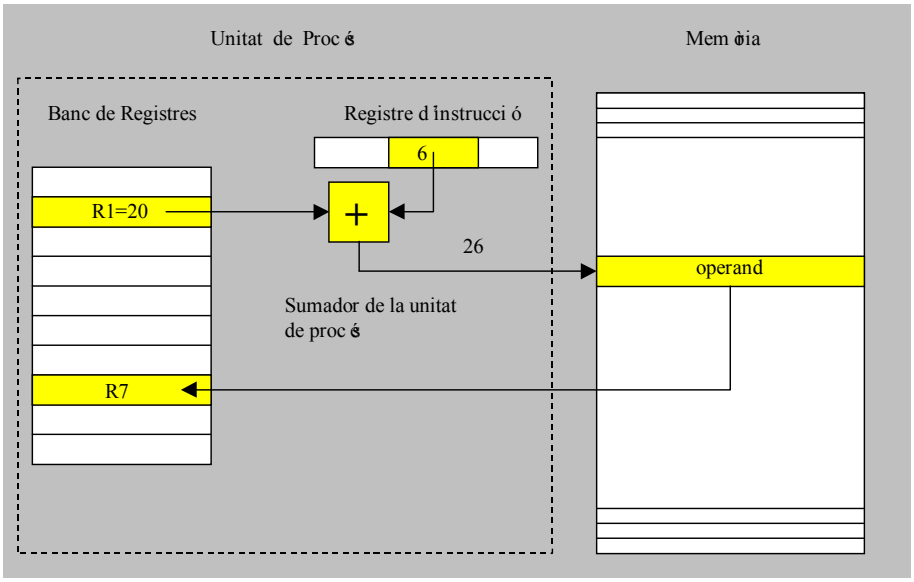


Mode base + desplaçament (també anomenat mode directe relatiu a registre base)

Aquest mode també permet d'especificar un operand que està guardat en una posició de memòria.

L'adreça on és l'operand (anomenada *adreça efectiva*) es calcula com la suma de dos nombres: una *adreça base* continguda en un registre (anomenat *registre índex*) i una quantitat entera anomenada *desplaçament*.

Per exemple, si suposem que el registre R1 conté el valor 20, la instrucció *MOV 6(R1),R7* escriu en el registre R7 el contingut de la posició de memòria que és a l'adreça 6 + 20 = 26, es a dir, R7:=M[26]. En aquest exemple, R1 fa el paper de registre índex i el nombre 6 és el desplaçament. En la figura següent es mostra l'execució d'aquesta instrucció.



Tal com veureu més endavant, en el cas particular de la Màquina Rudimentària, per calcular l'adreça efectiva s'utilitzen només els 8 bits de menys pes del registre índex (perquè les adreces són de 8 bits).

Els termes *adreça base* i *desplaçament* s'utilitzen de vegades de forma intercanviada: s'anomena desplaçament el contingut del registre índex. Aquest és el cas de la màquina Rudimentària.
Aquest canvi de nomenclatura no és important. El concepte important és que l'adreça efectiva es calcula com la suma d'una adreça base més un desplaçament.

Tipus d'instruccions

En aquest punt podeu començar a completar l'estudi amb el llibre o els apunts. Després de llegir la descripció de cada tipus d'instrucció que hi ha a continuació, heu d'estudiar al llibre el format detallat de cadascuna.

El llenguatge màquina de la Màquina Rudimentària està format per tres tipus d'instruccions.

Instruccions aritmètiques i lògiques

Aquestes instruccions executen operacions aritmètiques o lògiques entre dos registres del Banc de Registres o bé entre un registre i un operand immediat de 5 bits codificat en complement a dos. Escriuen el resultat en un registre i actualitzen els *indicadors de condició*, també anomenats *bits d'estat*.

La Màquina Rudimentària té les instruccions aritmètiques i lògiques següents:

- Sumar dos registres (ADD)
- Restar dos registres (SUB)
- Desplaçar un registre un bit a la dreta (ASR)
- *And* lògica de dos registres (AND)
- Sumar un registre i un operand immediat (ADDI)
- Restar un operand immediat d'un registre (SUBI)

Instruccions d'accés a memòria

Aquestes instruccions s'encarreguen de fer la transferència de dades entre la memòria i la UCP, i a l'inrevés.

La Màquina Rudimentària té dues instruccions d'aquest tipus:

- Moure un operand des de la memòria cap a un registre de la Unitat de Procés (LOAD).
- Moure un operand des d'un registre de la Unitat de Procés cap a la memòria (STORE).

Instruccions de salt

Aquest tipus d'instruccions són necessàries per a trencar el seqüenciament implícit del programa. Permeten saltar a la instrucció que hi ha a *l'adreça destí del salt* de forma incondicional o segons el resultat de la comparació de dos nombres enters.

La MR té les següents instruccions de salt:

- Saltar de forma incondicional
- Saltar si els dos operands són iguals
- Saltar si els dos operands són diferents
- Saltar si un operand és més gran que l'altre
- Saltar si un operand és més gran o igual que l'altre
- Saltar si un operand és més petit que l'altre
- Saltar si un operand és més petit o igual que l'altre

En aquest punt cal...

- Saber quines instruccions pot executar la Màquina Rudimentària i com funciona cadascuna d'aquestes instruccions.
- Comprendre com estan codificades les instruccions i per a què serveix cadascun dels camps del format de les instruccions.
- Saber traduir instruccions de codi màquina a assemblador i a l'inrevés. No cal saber fer-ho de memòria. Es pot consultar el llibre o els apunts.

3. Llenguatge assemblador de la Màquina Rudimentària

L'apartat A.2. conté un repàs de les instruccions del llenguatge assemblador de la Màquina Rudimentària.

A més de les instruccions, el llenguatge assemblador d'un computador disposa d'altres elements:

- Comentaris: permeten al programador d'explicar el significat del seu programa per tal de fer-lo entenedor a ell mateix i a altres persones que el puguin utilitzar posteriorment.
- Etiquetes: permeten d'identificar les adreces de memòria mitjançant una cadena alfanumèrica, de manera que la programació resulti més còmoda.
- Directives: permeten definir les zones de dades i les constants que usen els programes, i faciliten al computador algunes informacions addicionals necessàries perquè pugui executar el programa correctament.

Taula de símbols

La taula de símbols és una llista que construeix el programa assemblador per conèixer el valor de les etiquetes i constants. Cada etiqueta té associada l'adreça de la primera instrucció escrita després de l'etiqueta; les constants tenen associat el seu valor.

En aquest punt cal...

- Comprendre la funcionalitat de tots els elements del llenguatge assemblador.

Programa assemblador i simulador de la Màquina Rudimentària

A l'adreça ftp:
ftp://ftp.ac.upc.es/pub/archives/mr/win31/mr_20.zip
hi podeu trobar dues eines que ajuden a comprendre el funcionament de la MR:

- el **programa assemblador** tradueix de llenguatge assemblador a llenguatge màquina. Aquesta eina és útil per a comprovar si s'ha comès algun error en escriure el programa en llenguatge assemblador o en traduir manualment a llenguatge màquina.
- el **simulador** mostra gràficament el funcionament de la Unitat de Procés i de la Unitat de Control de la Màquina Rudimentària. És una eina molt útil per a

comprovar que s'entenen amb precisió tots els detalls del funcionament de la MR.

És convenient que llegiu ara l'apèndix A d'aquest apunts, que presenta una introducció a l'assemblador i al simulador de la MR, i feu els exercicis des de l'1.1 fins al 2.3 d'aquest apèndix.

En aquest punt cal...

- Saber usar el programa assemblador.
- Escriure programes en llenguatge assemblador i executar-los en el simulador. No cal entendre encara el contingut de les dues finestres de l'esquerra del simulador.

4. Disseny de la Unitat Central de Procés de la Màquina Rudimentària

En aquest capítol es construeix de forma completa el circuit digital que permet d'executar el joc d'instruccions que s'ha vist en el capítol anterior.

La Màquina Rudimentària és un computador pedagògic que disposa de totes les funcionalitats essencials que té qualsevol computador comercial. Els computadors reals són més complexos, en primer lloc perquè en una implementació física cal tenir en compte detalls que no cal considerar en un disseny sobre el paper, i en segon lloc perquè disposen de prestacions addicionals que els permeten d'optimitzar el rendiment. Però els components bàsics que permeten que un computador funcioni són els que conformen la Màquina Rudimentària.

La Màquina Rudimentària es pot veure com l'esquelet bàsic d'un ordinador real.

Per tant, si entenem el funcionament de la Unitat Central de Procés de la Màquina Rudimentària entenem també el funcionament de qualsevol processador real.

La Unitat Central de Procés consta de dos grans blocs: la Unitat de Procés i la Unitat de Control.

4.1. Disseny de la Unitat de Procés

Per a dissenyar la Unitat de Procés (UP) de la Màquina Rudimentària seguirem el procés següent:

a) **Enumerar les funcions que ha de satisfer la UP**.
b) **Per a cadascuna d'aquestes funcionalitats, escollir els blocs lògics que permetran implementar-les**.
c) **Connectar adequadament els diferents circuits parcials**.

a) Enumerar les funcions que ha de satisfer la UP. Estudiarem el que s'ha de fer en cadascuna de les fases d'execució d'una instrucció.

1. ***Fetch*** de la instrucció a executar. Per a dur a terme aquesta acció és necessari

   1.1. conèixer en tot moment l'adreça de memòria on hi ha la propera instrucció que s'ha d'executar,
   1.2. poder fer arribar aquest valor a l'entrada d'adreces de la memòria,
   1.3. disposar d'un lloc a la UP on guardar la instrucció que es llegeix de la memòria per ser executada.

2. ***Descodificació***: Aquesta acció requereix esperar el temps necessari perquè la Unitat de Control pugui analitzar la instrucció que s'està executant, de cara a donar unes ordres o unes altres a la Unitat de Procés. Per tant, no es necessari cap funcionalitat de la UP per a aquesta fase.

3. ***Cerca dels operands***. Els operands poden ser en tres espais d'adreces:

3.1. en un registre. Caldrà poder extreure (llegir) del Banc de Registres els operands indicats per la instrucció.
3.2. immediat. S'haurà de poder obtenir a partir de la instrucció en execució. L'immediat es codifica amb 5 bits i per tant cal estendre-li el signe, ja que la MR treballa amb nombres enters codificats en complement a 2 i 16 bits.
3.3. a memòria. En aquest cas, primer caldrà calcular l'adreça efectiva de l'operand, i després fer-la arribar al bus d'adreces de la memòria.

4. ***Execució i escriptura del resultat***. Per a aquestes fases es requereixen de la UP les funcionalitats següents:
   4.1. circuits capaços de realitzar totes les operacions aritmètiques i lògiques que apareixen en el repertori d'instruccions. També han d'ésser capaços de calcular els *indicadors de condició* (*bits d'estat*) a partir del resultat de l'operació
   4.2. camins que connectin els llocs on poden residir els operands font de les instruccions amb les entrades d'aquests circuits
   4.3. camins que connectin les sortides d'aquests circuits amb el Banc de Registres (en la MR, l'únic lloc on pot residir l'operand destí d'una instrucció)
   4.4. camins de connexió entre el Banc de Registres i la memòria, en tots dos sentits
   4.5. un lloc on guardar el valor dels indicadors de condició.

b) Per a cadascuna d'aquestes funcionalitats, escollir els blocs lògics que permetran implementar-les.

c) **Connectar adequadament els diferents circuits parcials**.

Aquest tipus de decisions cal prendre-les en dissenyar un computador real. Moltes vegades es tracta de decidir entre dues alternatives de disseny, una que permet d'aconseguir un més bon rendiment del computador però que requereix més circuiteria (i per tant és més cara i ocupa més espai al xip), i una altra que permet d'aconseguir un rendiment inferior però es pot implementar amb menys elements (i per tant és més barata i ocupa menys). És a dir, el dissenyador o arquitecte d'un computador es troba molt sovint davant una disjuntiva en la qual ha de cercar un compromís entre rendiment i preu.

Cal tenir present que la implementació de la Unitat de Procés no és determinista, sinó que es podria dur a terme de formes diverses. En realitzar el disseny s'han adoptat determinades decisions que li han donat la forma que té. Per exemple, es podia haver decidit que per a actualitzar els indicadors de condició en el cas de les instruccions LOAD es replicaria la circuiteria que els calcula, en comptes de fer que l'operand passi per dins la UAL. O es podia haver decidit que el Banc de Registres tingués dos ports de lectura, és a dir, dues entrades de selecció de lectura (*SL*) i dos busos de sortida de dades (*Dout*). D'aquesta manera se'n podrien llegir dos registres alhora.

En aquest punt cal...

- Comprendre quines funcionalitats ha d'oferir la Unitat de Procés.
- Conèixer amb detall els circuits que satisfan cadascuna de les funcionalitats.
- Entendre les interrelacions de tots els elements de la Unitat de Procés.

4.2. Disseny de la Unitat de Control

La Unitat de Control és un circuit seqüencial que governa la Unitat de Procés, és a dir, dóna en cada moment el valor adequat als senyals de control de la UP per tal que es duguin a terme ordenadament les accions requerides per a l'execució de les instruccions.

Els senyals de control que arriben a la Unitat de Procés són de tres tipus:

- Senyals de permís de càrrega dels diferents registres (Ld_PC, Ld_IR, Ld_RA, Ld_R@, Ld_RZ, Ld_RN).
- Senyals de selecció dels diferents multiplexors (PC/@, CRf, OPERAR).
- Permisos d'escriptura (ERd, L/E).

Per a donar en cada moment el valor correcte a aquests senyals, la Unitat de Control necessita saber quina instrucció s'està executant i, en cas que sigui una instrucció de salt, si es compleix o no la condició de salt. Per això a la Unitat de Control hi arriben els tres senyals IR_{15}, IR_{14} i Cond.

Cada fase d'execució d'una instrucció correspon a un estat del sistema seqüencial que és la Unitat de Control. El graf d'estats pot ésser simplificat posteriorment

Optimitzacions del diagrama d'estats de la Unitat de Control

Un dels objectius principals d'un arquitecte de computadors és que la màquina vagi tan ràpid com sigui possible. Les transicions entre els estats de la Unitat de Control es produeixen al ritme marcat pel rellotge del computador: a cada cicle es passa d'un estat al següent. Per tant, com menys estats s'hagin de recórrer per executar cada instrucció més ràpida serà l'execució dels programes.

Per això és convenient reduir al màxim el nombre d'estats de la Unitat de Control. Aquest és l'objectiu de les optimitzacions que es presenten en aquest apartat.

D'altra banda, el sistema seqüencial que implementa la Unitat de Control serà més senzill com menys estats hi hagi en el diagrama d'estats, i per tant el fet de reduir el nombre d'estats també contribueix a abaratir el cost de l'ordinador.

Altres maneres d'augmentar la velocitat d'un computador són reduir el temps de cicle i reduir el nombre d'instruccions dels programes.

En aquest punt cal...

- Entendre la seqüència d'accions necessàries per a l'execució de les diferents instruccions.
- Conèixer quins són els senyals que governen la Unitat de Procés i quina informació necessita la Unitat de Control per a donar-hi en cada moment el valor adequat.
- Comprendre el graf d'estats que descriu la Unitat de Control: les transicions i els valors dels senyals de sortida en cadascun dels estats.
- Entendre el funcionament de la Màquina Rudimentària, essent capaç d'expressar-lo mitjançant un cronograma.

5. Apèndix A: Introducció a l'assemblador i al simulador de la MR

En aquest apartat aprendreu a fer ús del simulador i l'assemblador de la Màquina Rudimentària. Amb aquestes eines podreu avaluar pel vostre compte els coneixements que aneu adquirint amb l'estudi d'aquest tema.

El simulador de la Màquina Rudimentària és una eina molt útil per a entendre'n el funcionament, i per tant el funcionament d'un computador real qualsevol. És capaç de simular l'execució de qualsevol programa escrit per a l'MR.

5.1 Instal·lació de les eines

Per a instal·lar el simulador de la MR al vostre PC seguiu els passos següents:

1. Carregueu el simulador de la MR per la xarxa des de la següent adreça ftp: ftp://ftp.ac.upc.es/pub/archives/mr/win31/mr_20.zip

2. Carregueu el programa de prova **prog** per la xarxa des de la següent adreça ftp: ftp://ftp.ac.upc.es/pub/archives/mr/win31/prog.zip

3. Descomprimiu el simulador i el programa de prova (fent servir la utilitat *WinZip*) i poseu-los en una carpeta anomenada SIM_MR.

4. Instal·leu el simulador fent doble clic a la icona del programa d'instal·lació **install.exe**. Us recomanem que instal·leu el simulador al directori C:\MR que us suggereix per defecte el programa d'instal·lació. És convenient que aquesta carpeta pengi directament del disc C:\, perquè haureu d'accedir-hi des del sistema operatiu MS-DOS per assemblar els programes i us resultarà més senzill.

5.2 L'assemblador

L'assemblador (*"ensamblador"* en castellà) és un programa que tradueix programes escrits en llenguatge assemblador a programes escrits en llenguatge màquina.

Els programes en assemblador s'han d'escriure en fitxers amb extensió ***.asm*** (per exemple *programa1.asm*). Com a resultat de l'assemblatge es genera un fitxer amb el mateix nom però amb extensió ***.cod***, que conté el programa traduït a llenguatge màquina de la MR.

El fitxer que contingui el codi del programa l'heu d'haver generat usant un editor de textos que guardi només el codi ASCII del text. Per exemple, si l'editeu en Word l'heu de guardar com a fitxer de text, ja que si no contindria altres caràcters de control a més del programa (també haureu de comprovar si Word li ha donat l'extensió ***.txt***, en aquest cas haureu de canviar-la per ***.asm***). Per a estalviar-vos feina, podeu fer l'edició amb el *notepad* o bé creant un fitxer nou amb el botó dret del ratolí, amb l'opció *Nuevo/Documento de texto*. Els fitxers han d'ésser al mateix directori on hi hagi el programa assemblador i el simulador.

Per a fer servir l'assemblador és pot treballar des d'una finestra de MS-DOS. Podeu obrir-ne una anant al menú de Windows *Inicio/Programas/MS-DOS*. En aquesta finestra escriviu els comandes següents:

C:\windows> **cd c:\mr**

C:\mr> **posten *nom_del_programa.asm***

Per a veure el codi màquina d'un programa cal executar el simulador amb la comanda:

C:\mr> **mr *nom_del_programa.cod***

i analitzar el contingut de la capsa que representa la memòria (està escrit en hexadecimal).

Un programa també es por assemblar *movent* l'icona del *programa.asm* sobre el executable de l'assemblador, anomenat ***posten***.

El simulador també es pot executar des de Windows fent doble clic sobre la icona **MR.exe**. Podeu obrir després el fitxer ***nom_del_programa.cod*** fent clic a la segona icona començant per l'esquerra (la que representa un fitxer).

Si intenteu editar el fitxer **.cod** us apareixeran a la pantalla símbols inintel·ligibles, per la raó següent: el que fan els editors de textos és interpretar els fitxers d'entrada com a conjunts de codis ASCII i dibuixar a la pantalla els símbols corresponents.

Per exemple, quan llegeixen la cadena de bits *01000001* entenen que és la codificació ASCII de la lletra 'A' i dibuixen a la pantalla aquest símbol. Però els bits del fitxer executable no formen codificacions ASCII, sinó instruccions de llenguatge màquina de la MR.

Per exemple, la cadena de bits *11011100* pot correspondre als 8 bits més alts de la instrucció ADD R4, R1, R3, però interpretada com a caràcter ASCII codifica el caràcter 220X (caràcter de subratllat "_"), que és el que apareixerà en pantalla si editem el fitxer.

5.3 El simulador

Un simulador és un programa que imita el comportament d'un aparell real (un computador, un avió etc.). Aquest programa té variables que corresponen a tots els elements de l'aparell. En cada moment calcula com variaria en l'aparell real l'estat de cada element a partir dels canvis que es donen al seu entorn, i modifica el valor de les variables en conseqüència. A més, generalment mostra per pantalla una aparença física semblant a la real: en els simuladors de vol, en la pantalla es veu el que es veuria des del lloc de comandament d'un avió (la semblança és encara més gran en els programes de realitat virtual).

Les funcions més importants del simulador de l'MR són:

- ***Cicle***: execució d'un cicle de rellotge. El simulador fa les modificacions corresponents a l'execució d'un cicle de rellotge i es queda aturat a l'espera d'una nova ordre. Aquesta funció s'invoca en prémer F7, la tercera icona o l'opció *Ciclo* del menú *Ejecutar*.
- ***Step***: execució d'una instrucció completa. El simulador recorre al graf d'estats de l'Unitat de Control els cicles corresponents a l'execució de la instrucció en curs i es queda aturat a l'espera d'una nova ordre. Aquesta funció s'invoca en prémer F8, la quarta icona o l'opció *Step* del menú *Ejecutar*.
- ***Run***: s'executa el programa des del punt on s'estigui aturat fins al final del programa o fins al proper punt d'aturada (*breakpoint*). Aquesta funció s'invoca en prémer F9, la cinquena icona o l'opció *Run* del menú *Ejecutar*.
- ***Reset***: el simulador torna a la situació en què estava en carregar el programa. Aquesta funció s'invoca en prémer F10, la sisena icona o l'opció *Reset* del menú *Ejecutar*.
- ***Watch***: permet de veure el contingut d'una o més posicions de memòria. Per a especificar la posició inicial se'n pot donar l'adreça o el nom de la variable que hi ha guardada. Aquesta funció s'invoca en prémer F4, la vuitena icona o l'opció *Watch* del menú *Debug*.

Al menú *Ayuda* d'ajuda del simulador hi ha una facilitat que n'explica el funcionament en detall.

- ***Breakpoint***: permet de definir un o més *punts d'aturada* en el programa. Un punt de trencament serveix perquè s'aturi la simulació en usar l'ordre *Run*, i és útil per a saltar-nos una seqüència d'instruccions de les quals ja hem estudiat el comportament o per a analitzar l'acció d'unes quantes instruccions en conjunt (per exemple, el cos d'un bucle).

5.4 Exercicis proposats

El programa de prova sobre el qual s'han proposat els exercicis (**prog.asm**), que heu baixat a través de la xarxa, és el següent:

```
     V:  .dw    0, -45, 4, -6, 78
suma:  .rw    1
       .begin inici
 inici:  AND   R0, R3, R3
         ADD   R0, R0, R4
         ADDI  R0, #5, R2
bucle:   SUBI  R2, #0, R0
         BEQ   fi
         LOAD V(R3), R1
         ADD   R1, R4, R4
         SUBI  R2, # 1, R2
         ADDI  R3, # 1, R3
         BR    bucle
   fi:   STORE   R4, suma(R0)
         .end
```

No obstant, podeu fer exercicis equivalents amb programes que hagueu escrit vosaltres mateixos (per exemple, els que resulten de la resolució dels problemes proposades en aquest tema).

1. Llenguatge assemblador i llenguatge màquina

1.1. Llegiu el programa. Què fa? Especifiqueu la missió de cada registre i de cada posició de memòria.

1.2. Traduïu-lo a mà a llenguatge màquina, construint la taula de símbols.

1.3. Invoqueu l'assemblador de la MR perquè el tradueixi a llenguatge màquina o utilitzeu directament el programa **prog.cod**. Us recomanem que l'assembleu vosaltres mateixos.

1.4. Compareu els dos codis binaris obtinguts. Hi ha cap diferència? A què és deguda?

2. Execució dels programes

Carregueu al simulador el programa de prova (seleccioneu l'opció *file*, la primera icona de l'esquerra).

2.1. Estudieu el contingut de la memòria abans de començar l'execució del programa:

a) Analitzeu i justifiqueu el contingut de les adreces 03h – 07h, tal com es mostren en la caixa que representa la memòria a la pantalla.

b) Feu un *watch* d'aquestes mateixes posicions de memòria (F4 o vuitena icona). Expliqueu les diferències que es produeixen respecte de l'exercici anterior.

2.2. Executeu el programa sense aturades fins al final (*run*, F9 o cinquena icona). S'ha produït cap variació en el contingut de la memòria? Quina?

2.3. Reinicialitzeu el simulador (*reset*, F10 o sisena icona). Poseu un punt d'aturada (*breakpoint*, F2 o novena icona) a l'última instrucció del bucle, executeu el programa (F9) aturant-vos després de cada iteració i anoteu les variacions que es produeixen als registres R1 – R4 en cadascuna.

a) Quan es donarà l'última volta al bucle? Per què?

b) Què passaria si traguéssim la instrucció ADDI R0,#5,R2 del programa?

3. Funcionament de la Màquina Rudimentària

3.1. Reinicialitzeu el simulador (F10), poseu un punt d'aturada a l'adreça 09h (F2) i executeu el programa fins a aquest punt (F9). Per a cadascuna de les instruccions que hi ha a les adreces 09h – 0Ch, responeu les preguntes següents:

a) Quina és la funció d'aquesta instrucció?

b) Quina informació d'entrada necessita la instrucció per a executar-se? On es troba?

c) Predieu quins registres de la Unitat de Procés (UP) modificaran el seu contingut després d'executar la instrucció. Comproveu-ho executant-la (*step*, F8 o quarta icona). Si us havíeu equivocat, esbrineu per què.

3.2. Reinicialitzeu la MR (F10) i estudieu el comportament de la Unitat de Procés (UP) en cada estat. Per a fer-ho, comenceu per l'estat inicial (*fetch*) i aneu avançant cicle a cicle fins que hàgiu passat per tots. Amb la simulació aturada en cadascun dels cicles, responeu les preguntes següents:

a) Mireu quins dels busos estan pintats en blanc, i expliqueu per què.

b) Predieu el valor de tots els components de la UP a l'estat següent. Comproveu-ho avançant un estat (F7). Si us havíeu equivocat, esbrineu per què.

3.3. Repetiu l'exercici 3.2 reinicialitzant la simulació i prement dues vegades seguides F9 (o la cinquena icona): s'executaran uns quants cicles i la simulació s'aturarà en un punt aleatori. D'aquesta manera, haureu de saber respondre les preguntes sense tenir la informació de què ha succeït anteriorment.

3.4. Seguint el mateix procés que en l'exercici 3.2 (avançar cicle a cicle), estudieu ara el comportament de la UC. En cada cicle, responeu les preguntes següents:

a) Estudieu quins dels senyals de la UC intervenen en les accions que s'estan duent a terme en aquest cicle. Expliqueu el perquè.

b) Justifiqueu el valor de tots els senyals de la UC.

c) Predieu quin serà l'estat següent. Comproveu-ho avançant un estat (F7 o tercera icona). Si us havíeu equivocat, esbrineu per què.

3.5. Repetiu l'exercici 3.4 però ara situant-vos en un estat aleatori (com en l'exercici 3.3).

6. Apèndix B: Conceptes bàsics de programació

Algorisme i programa

Un **algorisme** explica el procés necessari per a dur a terme una determinada tasca, especificant tots els passos que s'han de seguir i l'ordre en què s'han de fer.

Un cas típic d'algorisme és una recepta de cuina. Per exemple, l'algorisme per a fer pa amb tomàquet seria aquest:

1. llescar el pa fins que hi hagi prou llesques
2. per a tots els tomàquets que siguin necessaris,
   rentar-los
   tallar-los per la meitat
3. mentre quedin llesques per untar,
   fregar-les amb tomàquet
4. per a cada llesca,
   tirar-hi sal
   tirar-hi oli

Un **programa** és un algorisme escrit en un *llenguatge de programació*. Un **llenguatge de programació** té una sintaxi determinada, dissenyada per tal que els ordinadors la puguin entendre.

Hi ha llenguatges de programació *d'alt nivell* i llenguatges de *baix nivell*.

- Els **llenguatges d'alt nivell** utilitzen paraules i caràcters fàcilment intel·ligibles per les persones; són propers al llenguatge natural. Són llenguatges d'alt nivell el Fortran, el BASIC i el C, per exemple. Són independents de l'arquitectura de l'ordinador.

- Els **llenguatges de baix nivell** utilitzen símbols fàcilment intel·ligibles pels ordinadors: 0s i 1s (bits). Per a programar en aquest llenguatges es necessari conèixer l'arquitectura de l'ordinador.

Normalment, els programadors escriuen els programes en un llenguatge d'alt nivell. Després, els programes són traduïts a un llenguatge de baix nivell que els ordinadors puguin entendre. El procés de traducció s'anomena *compilació*. Una vegada un programa està traduït a baix nivell, l'ordinador el pot *executar*, és a dir, seguir els passos especificats pel programa.

En aquests apunts es fa servir un llenguatge d'alt nivell que anomenem *pseudocodi*. No és usat per cap computador, però és molt semblant a molts llenguatges reals. La diferència és que està basat en el català, mentre que els llenguatges comercials estan basats en l'anglès.

Els elements bàsics d'un programa són les *variables* i les *sentències*.

Les **variables**, igual que en una fórmula matemàtica, són símbols o paraules que en cada moment tenen un valor determinat i sobre les quals es poden fer operacions.

Les **sentències** són instruccions o ordres que especifiquen les operacions que cal portar a terme sobre les variables.

Declaració de variables

Al principi d'un programa s'ha d'especificar quines variables s'utilitzaran. Cada variable té un *nom* i un *tipus*:

- el **nom** és una paraula identificativa.
- el **tipus** indica quina mena de valors pot prendre la variable. Els tipus elementals de variable són *caràcter* i *enter*.

La sintaxi per a la declaració de variables és la següent:

var nom1, nom2...: <*tipus*>;

En una mateixa línia es poden declarar diverses variables que tinguin el mateix tipus, separades per comes. Per exemple, un programa podria començar així:

programa
var lletra1, lletra2 : ***caràcter***;
var nombre1, nombre2 : ***enter***;
var i : ***enter***;

Això indica que el programa usarà les variables de nom *lletra1* i *lletra2*, que podran contenir només caràcters, i les variables de nom *nombre1*, *nombre2* i *i*, que podran contenir només valors enters.

La paraula "*programa*" indica que comença el programa i la paraula "*fprograma*" indicarà que s'ha acabat. La paraula "*var*" indica que a continuació es farà la declaració de variables. Són *paraules reservades*, és a dir, no es poden fer servir per a cap altra cosa (per exemple, no es pot declarar una variable amb el nom "var"). Els noms dels tipus també són paraules reservades. Les paraules reservades se solen escriure subratllades, en lletra cursiva o en negreta. En aquest tema s'escriuen en lletra cursiva i negreta.

Sentències bàsiques

En un programa, les sentències s'escriuen en l'ordre en què han d'ésser executades, cadascuna en una línia diferent. Normalment, al final de cada sentència s'escriu el caràcter ";".

Les sentències bàsiques que tenen tots els llenguatges de programació són les següents.

- **assignació**: serveix per a donar un valor a una variable. En alguns llenguatges es denota amb els caràcters "**:**=", i s'usa la sintaxi següent:

  nom_variable **:** = expressió;

  L'*expressió* pot ser un valor concret, una variable o una operació matemàtica sobre variables i/o valors. Seguint l'exemple anterior, es podrien escriure aquestes sentències:

  ```
  lletra1 : = 'a';
  nombre1 : = 20;
  nombre2 : =  nombre1 + 1;
  ```

  El resultat d'executar aquestes sentències és que la variable *lletra1* pren com a valor la lletra 'a', la variable *nombre1* pren com a valor el nombre 20 i la variable *nombre2* pren el valor 21.

- **condicional**: serveix per a executar un determinat conjunt de sentències o un altre, segons si es compleix o no una determinada condició. La sintaxi és la següent:

  ```
  si <condició>      llavors <grup_de_sentències_1>;
          si no <grup_de_sentències_2>;
  fsi
  ```

  La *condició* és una relació d'igualtat o desigualtat entre dues variables o entre una variable i un valor, i s'acostuma a escriure entre parèntesis. Si la condició es compleix s'executa el *grup_de_sentències_1*, i si no, aleshores s'executa el *grup_de_sentències_2*. Per exemple:

  ```
  si  (nombre1 = 10) llavors        lletra2 := lletra1;
                                    nombre2 := 1;
                     si no          lletra2 := 'b';
                                    nombre2 := 0;
  fsi
  ```

  Com que havíem assignat a la variable *nombre1* el valor 20, la condició no es compleix. Per tant, el resultat d'executar aquesta sentència serà que la variable *lletra2* prendrà el valor 'b' i la variable *nombre2* prendrà el valor 0.

- **iterativa**: serveix per a executar repetidament un cert conjunt de sentències fins que deixi de complir-se una determinada condició. La sintaxi és la següent:

  ```
  mentre <condició> fer
          grup_de_sentències;
  fmentre
  ```

  Aquesta construcció s'anomena ***bucle***. Si la *condició* es compleix, s'executa el *grup_de_sentències* (s'anomena *cos del bucle*) i a continuació es torna a avaluar la condició. Si continua essent certa es torna a executar el cos del bucle, i així repetidament fins que la condició sigui falsa. En aquest moment passarà a executar-se la sentència que vingui a continuació del *fmentre*.

  Per exemple:

```
mentre (nombre1 > 15) fer
        nombre1 := nombre1 – 1;
        nombre2 := nombre2 + 1;
fmentre
lletra1 := 'c';
```

Com que inicialment *nombre1* té el valor 20, la condició és certa i s'executaran les dues sentències del cos del bucle. Això mateix passarà cinc vegades, fins que *nombre1* hagi arribat al valor 15; es diu que es faran *cinc **iteracions** del bucle*. En aquest moment la condició deixarà d'ésser certa, i s'executarà la sentència:

lletra1 := 'c'.

Resum

En aquest tema s'estudien els llenguatges de baix nivell i l'arquitectura bàsica d'un computador a nivell digital.

Els conceptes presentats s'apliquen a un computador senzill, anomenat Màquina Rudimentària. Aquest computador té quatre modes bàsics d'adreçament: els modes registre directe, immediat, absolut i relatiu directe amb registre base. Quant a instruccions, la Màquina Rudimentària té sis instruccions aritmètiques i lògiques, set instruccions de salt (una de les quals és de salt incondicional) i dues instruccions d'accés a memòria: una per a carregar dades i una altra per a emmagatzemar-les.

La Unitat de Procés de la Màquina Rudimentària conté els components digitals bàsics de qualsevol computador real. La Unitat de Control governa la Unitat de Procés per tal que les instruccions s'executin de forma correcta, i es dissenya a partir d'un graf d'estats.

Glossari

Assemblador: és un programa que tradueix programes escrits en llenguatge assemblador a programes escrits en llenguatge màquina

Bits de condició: també anomenats indicadors de condició o bits d'estat. Mantenen al computador informació sobre l'execució d'instruccions. La MR té dos bits: Z i N. El bit Z indica si el resultat de l'última operació ha sigut 0. El bit N indica si el resultat de l'ultima operació ha sigut negatiu

Bus de control: conjunt de cables que transporta senyals de control entre el processador i la resta d'unitats

Bus de dades: conjunt de cables que transporta dades entre les diferents unitats del computador

Bus d'adreces: conjunt de cables que transporta adreces des del processador fins a les altres unitats

Compilador: programa encarregat de fer la traducció d'un llenguatge de programació d'alt nivell a un altre (generalment de baix nivell)

Espais d'adreces: llocs on poden estar emmagatzemades les dades que utilitza un programa. En aquest curs considerarem que són el banc de registres, el registre d'instruccions i la memòria

Estructura von Neumann: distribució en blocs funcionals que segueixen la majoria dels computadors. Els blocs principals són la *Unitat Central de Procés*, la *memòria* i la *Unitat d'entrada/sortida*

Fases d'execució d'una instrucció: cada una de les etapes en les que es divideix l'execució d'una instrucció:

- Fase de ***Fetch***: es porta la instrucció des de la memòria fins al processador
- Fase de ***descodificació***: el processador examina quina instrucció s'ha d'executar
- Fase de ***lectura d'operands***: el processador localitza les dades (*operands*) que requereix la instrucció
- Fase **d'*execució***: el processador fa els càlculs indicats per la instrucció
- Fase **d'*escriptura de resultats***: es guarda (si cal) el resultat d'aquests càlculs

Llenguatge Màquina: llenguatge de programació que els computadors són capaços d'entendre, i que està format tan sols per zeros (0) i uns (1) (bits)

Llenguatge Assemblador: traducció a mnemotècnics del llenguatge màquina que el fa més comprensible per a les persones

Memòria principal: bloc que emmagatzema els programes i les seves dades

Modes d'adreçament: maneres d'indicar la situació dels operands d'una instrucció

Seqüenciament implícit: es diu que un computador té *seqüenciament implícit* quan, després d'executar la instrucció que hi ha a l'adreça *i*, executa la que hi ha a l'adreça *i+1* (excepte si la instrucció executada és una instrucció de salt)

Simulador: és un programa que imita el comportament d'un aparell real (un computador, un avió etc.)

Unitat Central de Procés: també anomenada *processador*, és el bloc on es duu a terme l'execució dels programes.

Unitat de Control: bloc del processador que dóna a la Unitat de Procés les ordres pertinents perquè les instruccions dels programes s'executin correctament

Unitat d'Entrada/Sortida: bloc que connecta el computador amb l'usuari a través dels perifèrics, ja siguin d'emmagatzematge (discos, cintes, etc.) o de comunicació amb l'usuari (impressores, mòdems, teclats, pantalles, etc.

Unitat de Procés: bloc del processador que conté la circuiteria necessària per a la realització de les operacions que necessiten les instruccions dels programes

Bibliografia

Hermida, R.; del Corral, A.; Pastor, E.; Sánchez F. (1998). *Fundamentos de Computadores*. Madrid: Ed. Síntesis.